Książka stanowi wprowadzenie do nowoczesnej i ważnej problematyki rozpoznawania obrazów. Przedstawiono w niej zarówno teoretyczne aspekty kilkunastu metod rozpoznawania, jak i liczne praktyczne przykłady. Podane algorytmy obliczeniowe ułatwiają wykorzystanie omawianych metod rozpoznawania, a ujednolicona formalizacja opisu tych metod i zaproponowana klasyfikacja pozwalają na łatwą orientację w rozmaitych podejściach, zwykle opisywanych w literaturze osobno i bez możliwości ich wzajemnego porównania.

Książka przeznaczona jest dla studentów informatyki, automatyki, robotyki i elektroniki, a także dla naukowców i inżynierów interesujących się metodami rozpoznawania obrazów.

Rozpoznawanie obrazów

Współczesna
Nauka i Technika

Informatyka

Ryszard Tadeusiewicz

Mariusz Flasiński

Rozpoznawanie obrazów

Państwowe Wydawnictwo Naukowe Warszawa 1991

Projekt graficzny serii
Zygmunt Ziemka

Tytuł dotowany przez Ministerstwo Edukacji Narodowej

Redaktor
Anna Głażewska-Czuryło

Pracę do druku przygotował Marian Łakomy

ISBN 83-01-10558-5

PAŃSTWOWE WYDAWNICTWO NAUKOWE

Wydanie I. Ark. wyd. 11,5. Ark. druk. 13,5.
Papier offsetowy kl. III, 80 g, 61 × 86 cm.
Oddano do reprodukcji w lipcu 1991 r.
Druk ukończono w sierpniu 1991 r. Zam. nr 1426.

ZAKŁ. GRAF. WYD. NAUK. ŁÓDŹ, UL. ŻWIRKI 2

SPIS RZECZY

1. WPROWADZENIE

Teoria rozpoznawania obrazów rozwinęła się na świecie wraz z badaniami związanymi ze *sztuczną inteligencją*. Jest to nowa gałąź informatyki, budząca wiele nadziei i jeszcze więcej kontrowersji. Zasadnicza idea badań nad sztuczną inteligencją sprowadza się do tego, by za pomocą urządzeń automatycznych (głównie komputerów) uzyskiwać działania maszyn podobne do tych, jakie realizuje człowiek za pomocą swojej inteligencji. Zakres działań specjalistów od sztucznej inteligencji zmienia się wraz z rozwojem informatyki, gdyż aktualnie wiele zadań – do niedawna uważanych za bezwarunkowo wymagające inteligencji – zdewaluowało się na skutek rozwoju techniki. Wynika to z braku precyzji określenia *inteligencja*, które – choć na pozór oczywiste – sprawia kłopoty przy próbach takiego zdefiniowania, by definicja nadawała się zarówno do oceny osobowości określonego człowieka, jak i do opisu działania programu komputerowego.

Przykład. W szkole za inteligentnego uważa się ucznia, który szybko i trwale zapamiętuje podawane wiadomości. Jednak w odniesieniu do systemów technicznych kryterium takie nie daje się zastosować, gdyż komputerowe bazy i banki danych zdolne są do trwałego i szybkiego zapamiętywania milionów informacji – a mimo to ich głupota doprowadza do łez wielu użytkowników. Tak więc inteligencja ucznia i „inteligencja" komputera – to najwyraźniej dwie różne rzeczy. Podobnie zawodne bywają kryteria przywiązywane zwyczajowo do określonych zawodów lub rodzajów pracy. Przykładowo, kadrę urzędniczą zwyczajowo nazywa się inteligencją pracującą, a tymczasem programy zastępujące pracę urzędnika – na przykład systemy przetwarzania danych pełniące funkcje podobne do księgowej w banku – uchodzą za najprostsze i najbardziej prymitywne ze wszystkich programów. Nie wystarczy także odwoływać się do sformułowań na temat pracy „twórczej" i „nietwórczej", gdyż komputerowe programy wspomagające twórczość inżynierską (tzw. CAD), a nawet komponujące muzykę lub kreujące dzieła plastyczne – nie są zaliczane do sztucznej inteligencji. Przykłady można mno-

żyć, odkrywając nawet swoistą hipokryzję w naszych ocenach, które są bardziej liberalne w odniesieniu do ludzi, niż w stosunku do komputerów. Rozważmy zagadnienie modnych ostatnio gier komputerowych. Jeśli człowiek błyskotliwie wygrywa partię szachową, to jesteśmy skłonni przypisać mu inteligencję – ba nawet mówi się niekiedy o geniuszu! A tymczasem gdy prosty program funkcjonujący na domowym komputerze ogrywa nas raz po raz – odmawiamy mu inteligencji.

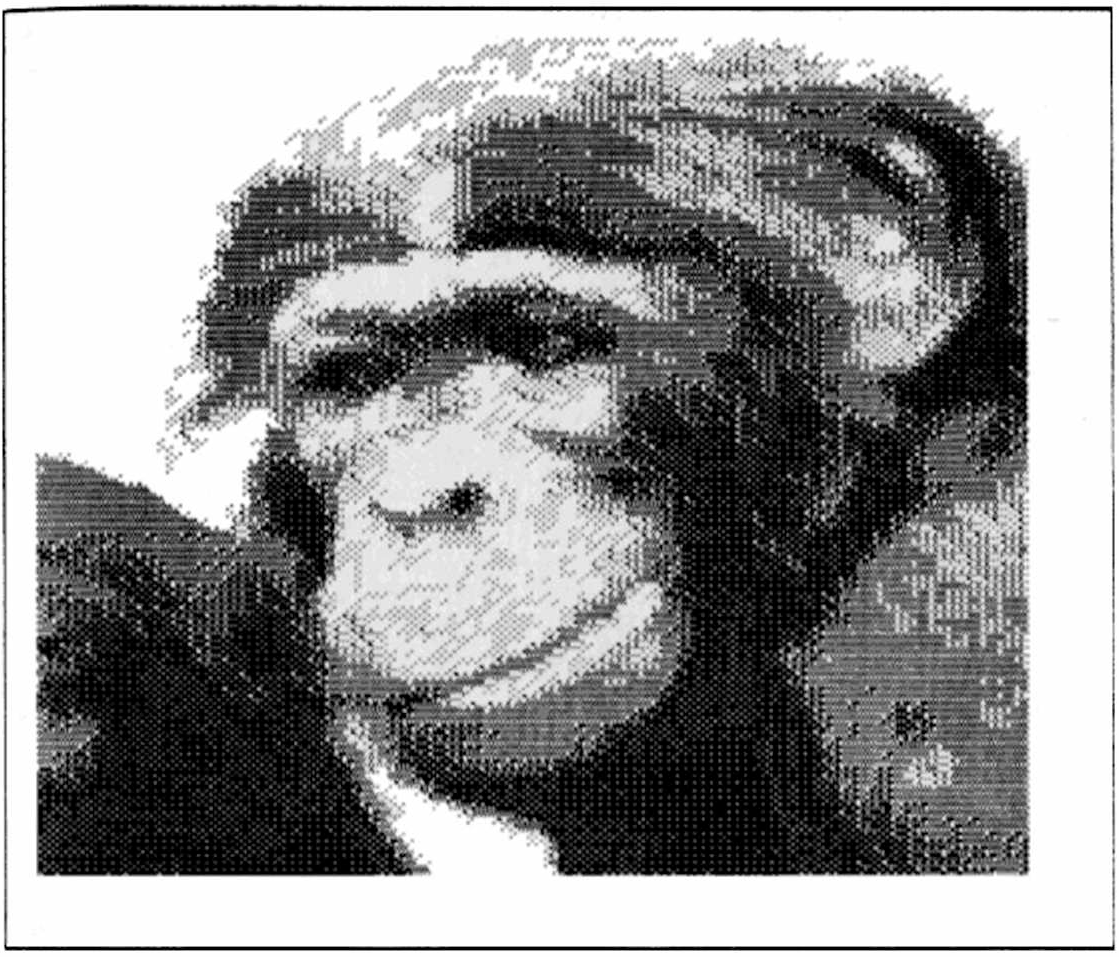
Jak z tego wynika, obszar sztucznej inteligencji ma bardzo słabo wytyczone granice, a niektóre klasyczne zagadnienia sztucznej inteligencji spadają do poziomu zadań szkolnych. Na przykład często wymieniana we wczesnych pracach z tej dziedziny łamigłówka o nazwie wieże Hanoi, *kamień probierczy* wielu programów sztucznej inteligencji, daje się dziś rozwiązać za pomocą programu zawierającego kilka linijek napisanych w języku LOGO, używanym w szkołach podstawowych. Jednak dla wszystkich specjalistów zajmujących się sztuczną inteligencją jedno nie ulega wątpliwości: zagadnienia rozpoznawania są nadal jednym z centralnych problemów tej dziedziny.

Polska nazwa *rozpoznawanie obrazów* niezbyt precyzyjnie odwzorowuje zakres tematyczny omawianego zagadnienia([1]) i z tego powodu wymaga pewnego komentarza. Pojęcie *obraz* zazwyczaj dość jednoznacznie kojarzy się z dwuwymiarową ilustracją (rys. 1.1) lub z trójwymiarową sceną, natomiast w nazwie dyskutowanej w książce dziedziny nauki musi być traktowane znacznie szerzej.

Przykład. Obrazem podlegającym rozpoznawaniu może być zarówno litera rękopisu lub odcisk palca, jak i stan pacjenta poddawanego postępowaniu diagnostycznemu, a także sygnał mowy (rys. 1.2), przebieg elektrokardiogramu (rys. 1.3) lub geofizyczny opis odwiertu złoża roponośnego. Przykładów jest zresztą bez liku: opis działania hydrauliki podwozia może być traktowany jako obiekt w zadaniu, w którym obrazami są decyzje o dopuszczeniu samolotu do kolejnego startu lub o skierowaniu go do remontu. Stan powierzchni chwytanego detalu może być obrazem dla systemu sensorycznego robota sortującego elementy na taśmie montażowej. Zbiór parametrów ekonomicznych opisujących gospodarkę przedsiębiorstwa może być rozpatrywany jako opis pewnego obiektu,

---

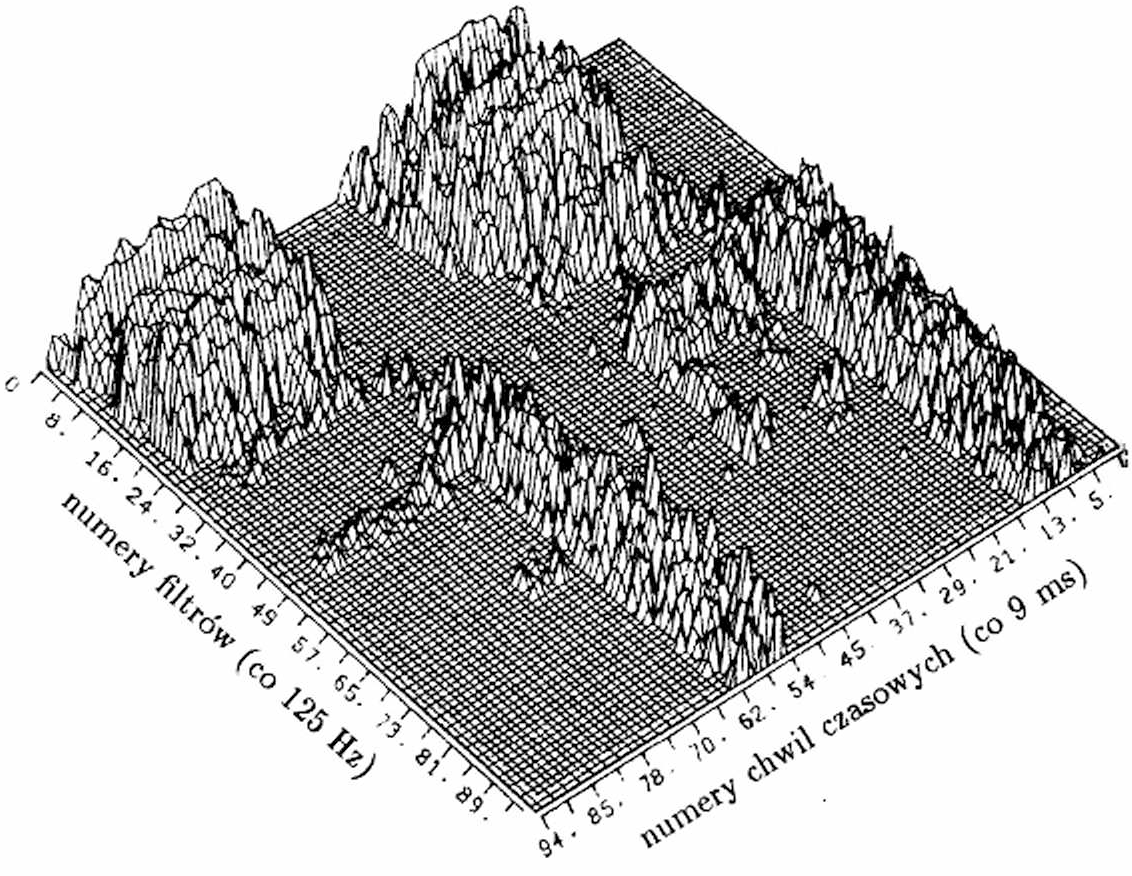
([1]) Znacznie korzystniejszy jest, powszechnie używany w literaturze dotyczącej rozpoznawania, anglojęzyczny termin *pattern recognition*, oznaczający rozpoznawanie wzorców.



Rys. 1.1. Wprowadzony do komputera obraz może być rozważany jako podlegający rozpoznawaniu obiekt. Jednak w dziedzinie *rozpoznawania obrazów* nie tylko takie obrazy są rozważane

co pozwala na automatyczne zaliczenie go do obrazu zakładów rozwojowych, wartych inwestowania lub stojących na progu bankructwa.

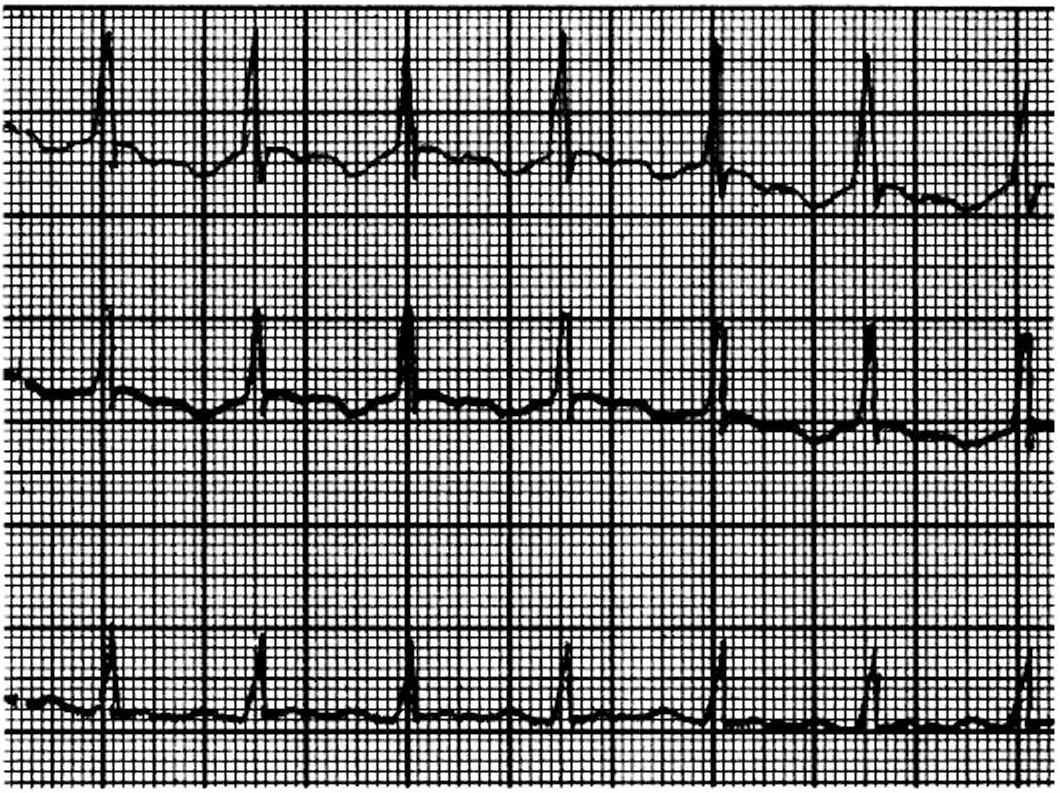
Ogólnie w zadaniu rozpoznawania obrazów chodzi o *rozpoznawanie przynależności rozmaitego typu obiektów (lub zjawisk) do pewnych klas.* Rozpoznawanie to ma być prowadzone w sytuacji *braku apriorycznej informacji* na temat reguł przynależności obiektów do poszczególnych klas, a jedyna informacja możliwa do wykorzystania przez algorytm lub maszynę rozpoznającą jest zawarta w *ciągu uczącym*, złożonym z obiektów, dla których znana jest prawidłowa klasyfikacja.



Rys. 1.2. Rozpoznawanymi obiektami mogą być wypowiedzi w języku naturalnym. Na rysunku widoczne jest widmo amplitudowo-częstotliwościowo-czasowe wyrazu *serce*. Zapis taki jest rozpoznawanym obiektem, zaś *obrazem* jest ogół wszystkich wypowiedzi tego wyrazu – niezależnie od tego, jakim głosem zostały wypowiedziane i bez względu na warunki nagrania

Przykład. W większości zadań automatyzacji diagnostyki medycznej (często rozważanych jako typowe zastosowanie rozpoznawania obrazów) punktem wyjścia do prowadzonych rozważań są kartoteki szpitalne. Zawierają one historie choroby wielu pacjentów, dla których właściwe diagnozy są (ex post) dobrze znane: wiadomo bowiem, jak tych pacjentów leczono i czym się to skończyło ...

Z teoretycznego punktu widzenia fakt braku apriorycznej wiedzy na temat algorytmu rozpoznawania powoduje, że zadanie rozpoznawania ob-



Rys. 1.3. Przykładem obiektu podlegającego rozpoznawaniu może być pokazany na rysunku elektrokardiogram. *Obiekt* taki może być zakwalifikowany do obrazów: *zapis prawidłowy, niedotlenienie komór, stan przedzawałowy* itp.

razów jest istotnie interesujące: maszyna powinna *uczyć się* rozpoznawania na podstawie przedstawionych przykładów. Zasadniczym elementem tego uczenia jest *uogólnianie*: komputer otrzymuje przykłady tylko *niektórych* obiektów (należących do ciągu uczącego), natomiast orzekać musi (w czasie rozpoznawania) o *wszystkich*. Z tego powodu rozpoznawanie obrazów jest *poligonem*, na którym powstają metody przekazywania urządzeniom automatycznym wszystkich tych możliwości i umiejętności, którymi człowiek dysponuje bez introspektywnej świadomości i wiedzy *jak to się robi*. O ile bowiem przekazanie maszynie umiejętności liczenia lub logicznego wnioskowania łączy się z wykorzystaniem znanych *algorytmów* tych czynności, o tyle liczne inne umiejętności człowieka, w tym umiejętności rozpoznawania, bazować muszą na prezentacji pokazów, gdyż algorytm jest nieznany. Jak wspomniano, konieczność bazowania przy rozpoznawaniu na *uogólnia-*

niu doświadczeń zebranych w postaci ciągu uczącego powoduje, że dziedzina rozpoznawania jest bardzo atrakcyjna z teoretycznego punktu widzenia. Równocześnie jednak fakt ten powoduje bardzo duże komplikacje praktyczne i sprawia, że zastosowania rozwiniętych już teorii są nieliczne, a efekty użytkowe – wciąż ograniczone.

Obiekty podlegające rozpoznawaniu mogą mieć różny charakter. Kilka przykładów już wymieniono, natomiast dalsze prezentowane są w kolejnych rozdziałach. *Zunifikowane* podejście do metod rozpoznawania jest możliwe dzięki temu, że jako wstępny etap procesu wymienia się *pomiar cech opisujących rozpoznawane obiekty*, zaś dalsze postępowanie polega na analizowaniu tych cech.

Przykład. Automatyczne rozpoznawanie sygnału KTG, będącego zapisem elektrycznej aktywności serca płodu w czasie porodu, prowadzone jest zwykle na podstawie oceny jego amplitudy w specjalnie wybranych pasmach częstotliwości. Wartości tych amplitud stanowią zatem cechy przy rozpoznawaniu KTG, a ich pomiar jest pierwszym etapem procesu rozpoznawania.

Wydobywane cechy są uzależnione od rodzaju rozpoznawanych obiektów, są więc zależne od konkretnego zadania i nie mogą być wybrane *raz na zawsze*.

Przykład. Podczas rozpoznawania mowy cechami są parametry akustyczne analizowanego sygnału: lokalizacja i czasowa zmienność tak zwanych formantów (częstotliwości rezonansowych w narządach mowy) oraz wartości momentów widmowych. Cechy te, bardzo skuteczne przy rozpoznawaniu mowy, są *całkowicie nieprzydatne* przy rozpoznawaniu innych sygnałów – nawet podobnych do mowy sygnałów dźwiękowych (np. w wibroakustycznej diagnostyce maszyn).

Po określeniu cech mamy jednak sytuację jednakową dla wszystkich zadań rozpoznawania: obiekt (dowolny) jest opisany zbiorem wartości swoich cech i na tym zbiorze wartości można dokonywać obliczeń w celu podjęcia decyzji o przynależności obiektu do określonej klasy. Teoria rozpoznawania obrazów dostarcza licznych metod podejmowania decyzji na podstawie określonych zestawów cech, nie daje natomiast podstaw do racjonalizacji wyboru samych cech, który pozostaje na ogół domeną *intuicji* konstruktora, tworzącego urządzenie lub algorytm do rozpoznawania. Fakt ten należy brać pod uwagę przy wszelkich próbach oceny metod rozpoznawania, gdyż wprowadza on element *arbitralności* do porządnie (w innych punktach) sformalizowanej teorii.

Przykład. W szczególnie często rozważanym zadaniu automatycznego rozpoznawania pisanych tekstów (dokładniej – liter i cyfr pisanych ręcznie lub maszynowo) opisuje się w literaturze kilkanaście alternatywnych zestawów cech, gwarantujących poprawne rozpoznawanie. Cechy te opisują kształty liter lub ich rzutów, rozmieszczenie punktów informatywnych (skrzyżowań, rozgałęzień i zakończeń linii), topologię zamkniętych i otwartych fragmentów konturów, obecność lub brak linii o określonym nachyleniu w określonych ćwiartkach prostokąta opisanego na rozpoznawanym znaku itp. Granicą poszukiwań jest tu jedynie limit wyobraźni twórców nowych metod, zaś ogromne zapotrzebowanie na układy OCR([2]) powoduje, że stale patentowane są nowe pomysły.

Warto dodać, że kiedy cechy są już wybrane (wytypowane), wówczas mogą być przedmiotem oceny i wartościowania z punktu widzenia ich przydatności w procesie rozpoznawania. Istnieją i są opisane w literaturze liczne metody selekcji informacji w systemach rozpoznających; często zresztą ten etap oceny i wartościowania stanowi najistotniejszy praktycznie efekt zastosowania metod rozpoznawania. Jednak taka analiza i selekcja możliwa jest jedynie *ex post*, kiedy jakieś cechy wstępnie zaproponowano, a potem poddano weryfikacji. Brak natomiast, co warto podkreślić, metod kreatywnych, pozwalających na automatyczne wygenerowanie *propozycji* cech. Propozycje takie muszą zawsze pochodzić od człowieka, znawcy specyfiki konkretnego zastosowania.

Przykład. Pomysłodawcą przy wyborze cech dla rozpoznawania odcisków palców musi być doświadczony daktyloskop, zaś cechy, na których ma się opierać automatyczne diagnozowanie raka trzustki musi podać wytrawny internista. Niepowodzenia opisywanych w literaturze prób stosowania metod automatycznego rozpoznawania obrazów do lokalizacji złóż roponośnych spowodowane były, jak się wydaje, faktem nieobecności wśród cybernetyków i informatyków opracowujących metodę – przynajmniej jednego geologa.

Niekiedy możliwe i celowe jest transformowanie jednych cech (opisujących rozpoznawane obiekty) w inne, możliwe do obliczenia na podstawie określonych reguł **transformacji cech**. Nowe cechy mogą pozwalać na łatwiejsze rozpoznawanie obrazów i ten fakt decyduje zazwyczaj o ich użyciu. Jednak reguły transformacji wykryte *przy okazji* dążenia do optymalizacji procesu rozpoznawania pozwalają na ogół wzbogacić wiedzę o rozpoznawa-

---

([2]) Optical Character Reader (optyczny czytnik znaków) to handlowa nazwa tych urządzeń.

nych obiektach – co także bywa wykorzystywane w praktyce jako *uboczny* efekt technik rozpoznawania.

Przykład. Często proponowanym przykładem transformacji cech jest przekształcenie Karhunena-Loevego, prowadzące do wydobycia najbardziej informatywnych i wzajemnie nie skorelowanych cech rozpoznawanych obiektów [3]. Dzięki zastosowaniu tego przekształcenia możliwe jest – w niektórych zadaniach – operowanie w czasie rozpoznawania kilkoma zaledwie, powstającymi po transformacji, składowymi kanonicznymi, zamiast kilkudziesięcioma cechami pierwotnymi.

Na podstawie cech, poddanych uprzednio selekcji i transformacjom lub wykorzystywanych bez żadnego wstępnego *preparowania*, trzeba podejmować *decyzje* o przynależności nieznanych obiektów do wyróżnionych obrazów (lub, ujmując to samo inaczej, do ustalonych klas). Możliwe są tu podejścia dwojakiego rodzaju: całościowe albo strukturalne.

Podejście całościowe polega na tym, że bierze się pod uwagę wszystkie cechy całego rozpoznawanego obiektu i podejmuje decyzję o jego przynależności w jednym etapie, w jednym akcie decyzyjnym. Oczywiście metody prowadzące do takiego podjęcia wymaganej decyzji mogą być rozmaite i dalej omówiono kilka spośród nich, porównując je ze sobą i wskazując na ich zalety w konkretnych sytuacjach. Opisano metody oparte na pojęciu *odległości* w przestrzeni cech i związanych z tym pojęciem intuicjach geometrycznych. Dyskutowano metody oparte na metodach *aprosymacji funkcji przynależności*, oferujące w określonych okolicznościach bardzo wysoką sprawność rozpoznawania. Możliwe jest podejście oparte na metodach probabilistycznych, nawiązujące do *statystyki*. Możliwości jest zresztą znacznie więcej. Przykładowo celowe jest niejednokrotnie wyróżnienie takich metod rozpoznawania, które mogą być stosowane także w przypadku znajomości zaledwie *niektórych* spośród wytypowanych cech. Naturalnie nie zawsze uda się poprawnie rozpoznać obiekt na podstawie znajomości tylko części cech, zatem w metodach tych próby rozpoznawania (połączone z uzupełnianiem wiedzy o cechach rozpoznawanego obiektu) trzeba wiele razy ponawiać, dlatego metody takie pozwalają znajdować rozwiązanie w wyniku **wieloetapowego** procesu, a nie jednorazowego aktu.

Podejście strukturalne jest odmienne: w rozpoznawanym obiekcie wyróżnia się najpierw określone *elementy* oraz ustala ich wzajemne *relacje*. Oczywiście przy wydzielaniu i identyfikowaniu elementów oraz przy

określaniu relacji wykorzystuje się – podobnie jak we wcześniej omówionych metodach – wybrane i pomierzone cechy. Jest to jednak dopiero etap wstępny, gdyż właściwe rozpoznanie dokonywane jest na podstawie strukturalnego opisu, uwzględniającego wszystkie wykryte elementy i wszystkie ustalone relacje. Metody podejmowania decyzji są w tym przypadku bardziej złożone i nawiązują (między innymi) do metod lingwistycznych podobnych do wykorzystywanych w językach i technikach programowania. W metodach tych oczywiście kluczową rolę odgrywa pojęcie *gramatyki*, ponieważ reguły tej gramatyki wyznaczają dopuszczalne formy obiektów podlegających rozpoznawaniu. W dalszych rozdziałach przedyskutowano najpopularniejsze gramatyki wykorzystywane specyficznie w rozpoznawaniu obrazów: gramatyki ciągowe, drzewowe i grafowe.

Przykład. W zadaniu rozpoznawaniu znaków alfanumerycznych relacjami są określenia definiujące bezwzględne i względne lokalizacje wyróżnionych cech. Na przykład określone skrzyżowanie linii może być „u góry" (relacja bezwzględna) lub „na lewo od zakończenia linii" (relacja względna).

Podejście strukturalne wydaje się na pozór bardziej złożone i mniej efektywne (przez dwuetapową analizę obiektów) od podejścia całościowego, jednak jego zaletą jest możliwość wykorzystania do analizy bardziej złożonych obrazów. Na przykład dokonując próby rozpoznawania złożonych scen, zawierających wiele obiektów, albo rozpoznając sygnał mowy w postaci całych, wielowyrazowych wypowiedzi – możemy zdecydowanie usprawnić proces rozpoznawania stosując podejście strukturalne. Jego użycie jest uzasadnione tym, że we wspomnianych zadaniach mamy do czynienia z problemami, których złożoność – przy próbach całościowego rozpoznawania – przekracza możliwości współczesnych systemów rozpoznających, a które można z powodzeniem rozwiązać poprzez zastosowanie podejścia strukturalnego. We wspomnianych zadaniach dość oczywiste jest także zdefiniowanie sposobu dekompozycji złożonego zadania na składniki elementarne, oczywisty jest także sposób wprowadzenia składników elementarnych oraz wyodrębnienie relacji.

Podejście strukturalne bywa także użyteczne w zadaniach rozpoznawania obrazów, w których nie występują w sposób jawny żadne konkretne obiekty, a podział wejściowych informacji na klasy oparty jest na pewnych cechach globalnych, trudnych do opisu i formalizacji.

Przykład. Do zagadnień tego typu należy problem rozpoznawania i klasyfikacji faktury rozważanych powierzchni. Do zagadnienia tego powrócimy przy omawianiu metod strukturalnych, w tym miejscu warto jedynie podkreślić odmienność tego zadania od innych typowych problemów rozpoznawania obrazów.

W literaturze dotyczącej problematyki rozpoznawania formułuje się także komplementarne – w stosunku do zadania *prostego rozpoznawania* – zadanie *grupowania*([3]), mające liczne praktyczne zastosowania, szczególnie w ekonomii, medycynie lub technice przetwarzania sygnałów. Zadanie to najłatwiej wprowadzić jako *inwersję* zadania rozpoznawania. Przy rozpoznawaniu mamy *dany* zbiór klas oraz pojedynczy obiekt, którego *przynależność* do jednej z klas ma być ustalona i wykazana. W zadaniu klasteryzacji mamy *daną* jedynie zbiorowość obiektów, które zapewne dzielą się na jakieś klasy, przy czym liczba i charakterystyki klas nie są znane i powinny być dopiero *automatycznie wyznaczone*. Dopiero z tego podziału wyniknie przynależność określonych obiektów do poszczególnych klas.

Charakterystyczną współzależność zadań rozpoznawania i klasteryzacji pogłębia fakt, że znaczna część metod i technik wykorzystanych do rozpoznawania daje się adaptować dla potrzeb klasteryzacji i na odwrót.

Przykład. Szeroko znany pakiet ARTHUR, opracowany na Uniwersytecie Washingtion dla komputera CDC Cyber pozwala (zależnie od wyboru użytkownika) rozwiązywać zadania rozpoznawania lub dokonywać klasteryzacji danych. Ten pakiet wykorzystywano podczas badania właściwości omawianych w kolejnych rozdziałach metod rozpoznawania, a także służył jako punkt odniesienia przy ocenie efektywności niektórych nowych metod rozpoznawania.

Metody grupowania mają jednak swoją specyfikę, której wprowadzenie rozbiło by wewnętrzną spójność tej książki, a ponadto dla tych metod – w odróżnieniu od metod rozpoznawania – opracowano niedawno nowe podręczniki. Dlatego mimo bliskiego związku z zagadnieniami tu prezentowanymi, metody grupowania i analizy skupień pozostaną poza zakresem tej książki.

([3]) W literaturze światowej ustalił się dla tej dziedziny angielski termin *cluster analysis*, który bywa niekiedy spolszczany jako „klasteryzacja". Będziemy tę nazwę także stosowali, mimo jej niezbyt ładnego brzmienia w języku polskim, ze względu na krótszy i wygodniejszy zapis.

2. ZADANIE ROZPOZNAWANIA

2.1. Klasyfikacja jako punkt wyjścia do rozpoznawania

W poprzednim rozdziale wprowadzono zadanie rozpoznawania w sposób intuicyjny i opisowy. W celu konkretyzacji tego zadania i wprowadzenia poszczególnych metod rozpoznawania korzystne jest jednak dysponowanie ujęciem bardziej sformalizowanym. Takie właśnie ujęcie zostanie teraz przedstawione.

Oznaczając przez D zbiór obiektów lub zjawisk podlegających rozpoznawaniu, możemy przyjąć, że na zbiorze tym zdefiniowana jest relacja $K \subset D \times D$, będąca relacją równoważności. Relacja K określa rozbicie zbioru D na kolekcję klas równoważności $\{D^i\}$, odpowiadających poszczególnym obrazom. Relacji K nadamy nazwę *klasyfikacji*. W dalszych rozważaniach zakładamy jedynie jej *istnienie*, co jest potrzebne do prawidłowego sformułowania zadania rozpoznawania. Żadne bliższe charakterystyki klasyfikacji K nie są z góry narzucone, ponieważ konstruktor algorytmu lub urządzenia przeznaczonego do rozpoznawania nie zna (z założenia) bliższych własności tej relacji. Założenie to musimy wprowadzić w celu zapewnienia odpowiedniego poziomu ogólności prowadzonych rozważań, gdyż w większości niebanalnych zagadnień kryteria klasyfikacji są nieznane, zaś w pozostałych przypadkach ich znajomość czyni zadanie rozpoznawania trywialnym.

Oznaczmy przez L liczbę klas generowanych przez relację K, a zbiór indeksów klas – przez I. Wówczas

$$D = \bigcup_{i \in I} D^i, \tag{1}$$

$$\forall_{\mu,\nu\in I}\left[D^{\mu}\cap D^{\nu}=\emptyset\right], \tag{2}$$

$$\forall_{d^{\mu},d^{\nu}\in D}\left[\langle d^{\mu},d^{\nu}\rangle\in K\Rightarrow\exists_{i\in I}(d^{\mu}\in D^{i})\wedge(d^{\nu}\in D^{i})\right]. \tag{3}$$

Z opisu relacji K i zbioru I wynika istnienie odwzorowania

$$A : D \to I \tag{4}$$

o własnościach

$$\forall_{d\in D}\left[A(d)=i\ \equiv d\in D^{i}\right]. \tag{5}$$

Odwzorowanie A w pełni opisuje relację K, natomiast relacja K definiuje odwzorowanie A z dokładnością do permutacji zbioru indeksowego I. Z tego powodu uważać można, że odwzorowanie A – w odróżnieniu od istniejącej obiektywnie (z założenia) relacji K – zawiera pewien arbitralny składnik, związany z wyborem sposobu numeracji klas.

Przykład. W zadaniach diagnostyki medycznej, należących do „klasyki" rozpoznawania obrazów zbiór D utożsamia się ze zbiorem wszystkich rozważanych dolegliwości. Dolegliwości te nie są (obiektywnie) identyczne i dlatego musi się rozróżniać rozmaite choroby, co odpowiada klasyfikacji K. Natomiast przypisanie nazw chorobom, co odpowiada odwzorowaniu A, jest oczywiście arbitralne.

2.2. Zadanie rozpoznawania

W zadaniu rozpoznawania dąży się do tego, aby skonstruować algorytm realizujący odwzorowanie

$$A : D \to I \cup \{i_o\} \tag{6}$$

takie, aby pewna miara $Q(A,\hat{A})$, nazywana dalej *oceną jakości algorytmu rozpoznawania* $\hat{A}$, była minimalna. Jednoelementowy zbiór $\{i_o\}$ sybolizuje tu brak odpowiedzi (decyzja typu *nie wiem*). Wprowadzenie w odwzorowaniu $\hat{A}$ elementu i_o czyni zadanie rozpoznawania bardziej realistycznym: w praktyce często nie można ustalić prawidłowej decyzji z całą dokładnością, a znacznie lepiej jest, jeśli algorytm uzna, że nie potrafi rozpoznać określonego obiektu i zgłosi to specjalnym sygnałem, niż kiedy zgłoszony zostanie mylnie rozpoznany element.

Czasami przy definicji algorytmu $\hat{A}$ dopuszcza się *rozpoznania wariantowe*, to znaczy przyjmuje się sytucję, w której jako rozpoznanie akceptuje się dowolny *podzbiór* zbioru I. Warto zauważyć, że takie rozwiązanie też może być przydatne z punktu widzenia praktycznych zastosowań, ponieważ niekiedy wystarczy wskazać pewien podzbiór klas rozpoznawanych obiektów, aby osiągnąć zamierzony cel.

Przykład. Przy rozpoznawaniu mowy może być wystarczające stwierdzenie, że analizowana głoska jest szumowa, bez precyzowania, czy mamy do czynienia z S, SZ czy Ś, a przy diagnostyce medycznej może nas zadowalać, że podejrzenie raka może być wykluczone - chociaż dokładniejsze rozpoznanie choroby nie jest jeszcze możliwe.

Odwzorowanie $\hat{A}$ może być zatem opisane jako

$$\hat{A} : D \to 2^I, \tag{6a}$$

gdzie oznaczenie 2^I użyte jest (zgodnie z tradycją) do zapisu *zbioru wszystkich podzbiorów zbioru* I. Warto zwrócić uwagę, że w tym przypadku zbędne jest wprowadzanie elementu i_o, ponieważ zbiór 2^I zawiera (z definicji) zbiór pusty ($\emptyset \subseteq 2^I$), czyli właśnie brak rozpoznania. Odmowa rozpoznania może zresztą być przy takiej konwencji wyrażona na dwa sposoby: albo poprzez podanie jako rozwiązania zbioru pustego $\emptyset$, albo poprzez podanie jako rozwiązania całego zbioru I, co jest możliwe, jako że $I \in 2^I$.

2.3. Elementy składowe rozpoznawania

Odwzorowanie A jest realizowane jako założenie *trzech* odwzorowań

$$A = F \cdot C \cdot B, \tag{7}$$

przy czym *pierwsze* z nich

$$B : D \to X \tag{8}$$

będziemy nazywać recepcją, *drugie*

$$C : X \to R^L \tag{9}$$

oznacza obliczanie wartości tak zwanych *funkcji przynależności*, zaś *ostatnie* odwzorowanie, zapisywane jako

$$F : R^L \to I \cup \{i_o\} \tag{10}$$

lub

$$F : R^L \rightarrow 2^I \tag{10a}$$

oznacza proces podejmowania decyzji.

Przykład. W systemach wizyjnych robotów przemysłowych odwzorowanie B polega na wprowadzeniu potrzebnego obrazu do komputera (za pomocą kamery TV i przetwornika A/C) oraz na obliczeniu (za pomocą specjalizowanych procesorów) wybranych cech rozpoznawanych obiektów. Na podstawie tych cech określane są (za pomocą odpowiednich programów w nadrzędnym komputerze) miary podobieństwa do znanych wzorców (odpowiada to odwzorowaniu C) oraz podejmowana jest (przez sterownik robota) decyzja, który obiekt należy pochwycić (odwzorowanie F).

Warto dodać, że schemat ten nadaje się do opisu metod całościowego rozpoznawania, natomiast metody strukturalne, omawiane w dalszych rozdziałach książki, wymagają odmiennego podejścia, co zostanie we właściwym miejscu wyraźnie zaznaczone. Omówimy obecnie bliżej wprowadzone odwzorowania.

2.4. Recepcja i struktura przestrzeni cech

Początkowym elementem każdego algorytmu rozpoznającego jest pomiar cech wszystkich obiektów – zarówno wzorcowych, należących do ciągu uczącego (patrz dalej), jak i podlegających rozpoznawaniu

$$B : D \rightarrow X.$$

Określenie cech prowadzi do zamiany obiektów $d \in D$ w punkty pewnej przestrzeni. Symbol X w zacytowanym wzorze (8) oznacza właśnie tę *przestrzeń cech*. Jej struktura jest z reguły arbitralna i zdeterminowana głównie przez możliwości pomiarowe. Trudno bowiem zakładać uwzględnianie w rozpoznawaniu takich cech, których wartości nie potrafilibyśmy wyznaczyć!

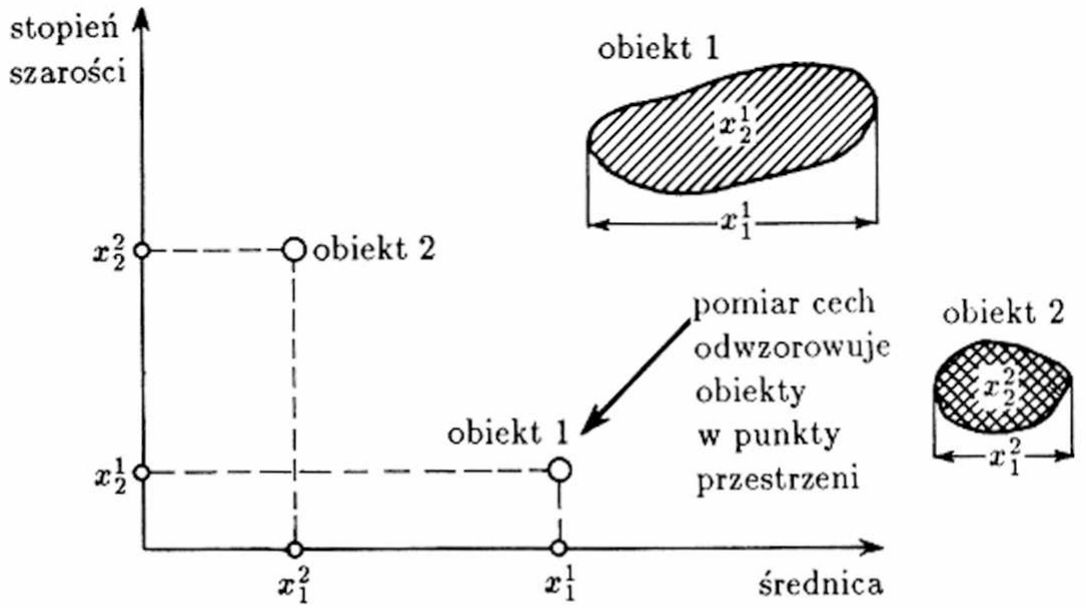
Zakładać będziemy, że elementami przestrzeni cech X są wektory n-elementowe

$$\underline{x} = \langle x_1, x_2, \ldots x_n \rangle \in X. \tag{11}$$

Składowe x_ν tych wektorów chętnie będziemy traktowali jako *liczby* $x_\nu \in \mathcal{R}$ określające ilościową *miarę* określonej cechy, co powoduje, że przestrzeń X traktowana będzie jako n-wymiarowa przestrzeń euklidesowa ($X \subseteq \mathcal{R}^n$). W takiej przestrzeni stosunkowo najłatwiej i w najbardziej naturalny sposób można będzie prowadzić wszystkie analizy i rozważania, dlatego przestrzeń ta traktowana będzie przez nas jako *model standardowy*.

Dla ilustracji prowadzonych rozważań przyjmować będziemy dodatkowo, że mamy do czynienia z przestrzenią dwuwymiarową ($n = 2$), aby można było odwzorowywać rozważane obiekty jako punkty na płaszczyźnie, zaś w niektórych przypadkach uciekać się będziemy nawet do „przestrzeni" jednowymiarowych, co oczywiście nie odpowiada na ogół rzeczywistemu zadaniu rozpoznawania, a służyć będzie głównie temu, by wygodnie zaprezentować pewne prawidłowości i współzależności na rysunku.

Przykład. Na rysunku 2.1 przedstawiono, w jaki sposób cechy obiektów wyznaczają współrzędne punktów w przestrzeni cech i jak dochodzi do odwzorowania obiektu $d^\mu \in D$ w punkt $\underline{x}^\mu \in X$, charakteryzowany przez współrzędne x_1^μ



Rys. 2.1. Rozpoznawane obiekty mogą być traktowane jako punkty w przestrzeni cech. Na rysunku pokazano przestrzeń cech, w której na osi poziomej odkładana jest średnica obiektu, a na osi pionowej – jego stopień szarości

---

(1) Dalsze szczegóły na ten temat – patrz: Dodatek 1.

i x_2^μ, interpretowane (w rozważanym na rysunku 2.1 przykładzie) jako średnica obiektu i jego stopień szarości.

Sytuacja, kiedy cechy mogą być interpretowane jako liczby, jest najkorzystniejsza, jakkolwiek nie jedyna. Cechami w ogólnym przypadku mogą być również pojedyncze *bity* sygnalizujące *obecność lub brak* określonej właściwości rozważanego obiektu, a także *symbole kodowe* (nazwy) określające wartości cech *porządkowych* lub wręcz *jakościowych* rozważanych obiektów([1]). Oczywiście w takim przypadku przestrzeń X nie może być traktowana jako euklidesowa, co jednak nie przeszkadza we wprowadzeniu pewnych intuicji geometrycznych.

Przykład. Na rysunku 2.2 pokazano przestrzeń cech opartą na danych binarnych, a na rysunku 2.3 – przestrzeń cech porządkowych i opisowych. Jak widać także i w przypadku tych cech możliwe jest utożsamianie obiektów z punktami w przestrzeni, chociaż przestrzeń ta nie jest tak gęsto wypełniona punktami, jak przestrzeń liczb rzeczywistych.

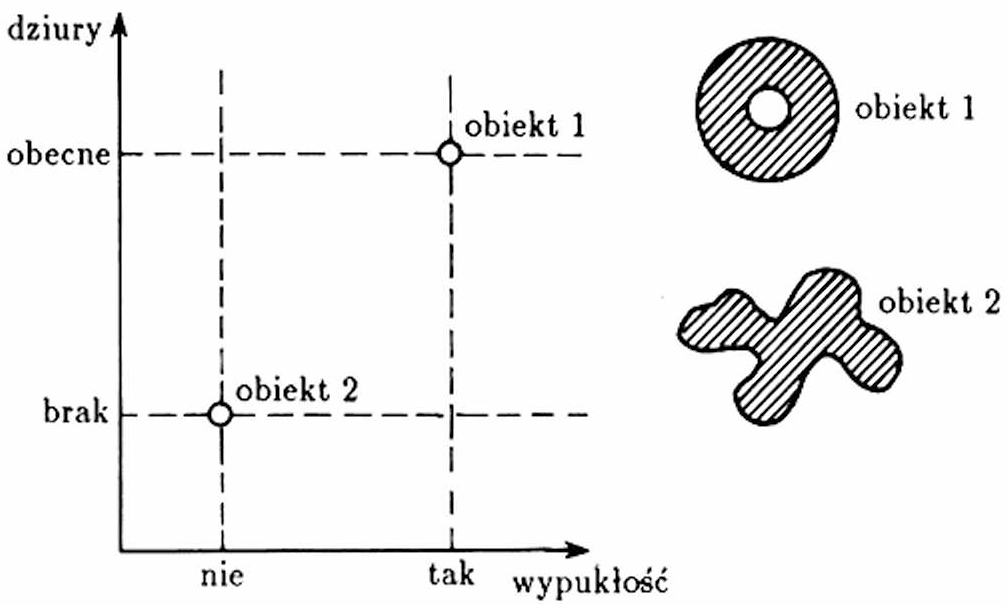
Rodzaj i własności wybranej przestrzeni cech bardzo silnie wpływają na dalszy tok procesu rozpoznawania. Jest to zupełnie zrozumiałe: obiekty $d \in D$ mają *potencjalnie* nieskończenie wiele cech. Odwzorowanie B prowadzące do n-wymiarowej ($n \ll \infty$) przestrzeni cech X związane jest zawsze z *utratą części informacji*, zatem jeśli utracona zostanie informacja istotna z punktu widzenia celów rozpoznawania, a w przestrzeni cech uwzględni się wyłącznie cechy mało ważne - to straty tej nie da się zrekompensować żadnymi późniejszymi wysiłkami.

Nie ustalono dotychczas żadnych ścisłych metod określania struktury przestrzeni cech i jej wybór ma w dużej mierze charakter heurystyczny i arbitralny, zależny od własności zbioru D oraz od pomysłowości twórcy algorytmu A. Zagadnienie to było już sygnalizowane w poprzednim rozdziale. W literaturze istnieje na ten temat jedynie kilka ogólnikowych wskazówek, na przykład szeroko znana jest *zasada Brawermanna*. Zasada ta głosi, że cechy x_ν muszą być tak dobrane, aby w przestrzeni cech X punkty x odpowiadające obiektom d należącym do jednej klasy ($d \in D_i$) grupowały się w postaci skupisk możliwie maksymalnie zwartych wewnętrznie i możliwie najbardziej oddalonych od podobnych skupisk dla innych klas([2]). Zasada

([2]) L.I. Rozonoer opisał tę zasadę obrazowo w ten sposób, że różnice pomiędzy klasami w przestrzeni cech muszą być innego rodzaju, niż różnica, jaka istnieje pomiędzy gąbką a nasączającą ją wodą.

ta jest jednak zbyt ogólnikowa, by mógł z niej rzeczywiście skorzystać twórca maszyny rozpoznającej.

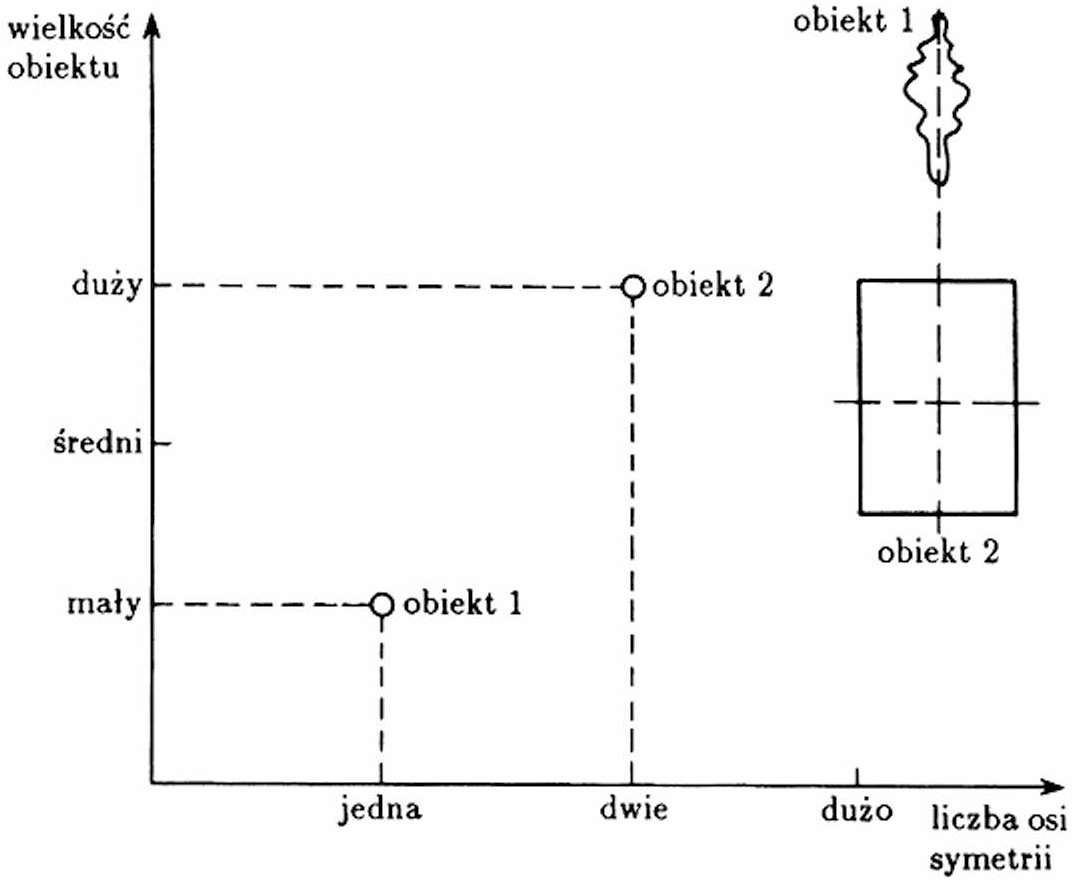
Ostatnio w zagadnieniach związanych z wyborem cech i optymalizacją struktury przestrzeni cech zaistniały nowe fakty. Spore szanse na rozwiązanie tego problemu wiązać można z bardzo intensywnie rozwijanym działem sztucznej inteligencji, poświęconym *systemom ekspertowym*, w ramach którego – być może – wypracowane zostaną także metody automatycznego wyboru cech w zadaniach rozpoznawania. Pewne nadzieje wiązać można także z powstającą obecnie dziedziną techniki, zwaną *inżynierią wiedzy*.



Rys. 2.2. W przestrzeni cech można umieszczać także obiekty opisane cechami binarnymi (oznaczającymi obecność lub brak określonej własności). Na rysunku przedstawiono przestrzeń cech, w której na osi poziomej oznaczono wypukłość linii konturowej obiektu, a na osi pionowej – obecność lub brak wewnętrznych konturów

Dotychczas stale zakładaliśmy, że wszystkie cechy x_j $(j = 1, 2, \ldots, n)$ są dostępne równocześnie. Tymczasem proces pozyskiwania cech może wiązać się z pewnymi trudnościami (na przykład kosztami) i zwykle jest rozłożony w czasie.

Przykład. W diagnostyce medycznej poszczególne badania, których wynikiem są kolejno wykrywane symptomy (rozważane tu jako cechy, będące podstawą rozpoznawania) są uciążliwe dla pacjenta i obciążające dla placówki dia-



Rys. 2.3. Przestrzeń cech może być oparta na cechach mających charakter kodów opisujących właściwości obiektów. Na osi poziomej odłożono kody odpowiadające liczbie osi symetrii obiektu (w skali: jedna, dwie, dużo), a na osi pionowej – kody odpowiadające wielkości obiektu (w skali: mały, średni, duży)

gnozującej. Jeśli zatem można podjąć decyzję na podstawie analizy tylko niektórych cech x_j $(j = 1, 2, \ldots, m)$, to wówczas można sobie (i pacjentowi) zaoszczędzić trudu określania symptomów $x_{m+1}, \ldots, x_n$, co może stanowić istotny zysk i zwykle przyspiesza rozpoznanie.

Formalna struktura odwzorowania B dla omawionego tu częściowego (kolejnego) określania cech może być w zasadzie rozważana jako identyczna ze strukturą podaną wzorem (8), gdyż różnica polega jedynie na liczbie wymiarów przestrzeni X. Jednak dla zaakcentowania odmienności rozważanego tu podejścia odwołamy się ponownie do zapisu o postaci 2^X, określającego (w tym przypadku) zbiór wszystkich podzbiorów (podprzestrzeni) wektorowej przestrzeni cech X. Wprowadzimy także wyróżnik e dla zazna-

czenia, że chodzi tu o rozpoznawanie *etapowe* i zapiszemy odwzorowanie B w zmodyfikowanej postaci wzoru (8)

$$B^e : D \to 2^X.$$

Jeśli określenie pierwszych m cech nie da wystarczającej podstawy do podjęcia wymaganej decyzji – można postępowanie diagnostyczne (lub inną procedurę rozpoznawania) kontynuować według tej samej zasady – albo mierząc od razu wszystkie pozostałe cechy, albo znowu poprzestając na ustaleniu wartości niektórych z nich. Do zagadnienia tego powrócimy w następnych rozdziałach.

2.5. Funkcje przynależności

Omówione odwzorowanie B może być traktowane jako samo tylko zbieranie danych o właściwościach rozpoznawanego obiektu $d \in D$. Natomiast kolejne odwzorowanie $C : X \to \mathcal{R}^L$ traktowane musi być znacznie poważniej. W odwzorowaniu tym chodzi o ustalenie pewnej miary podobieństwa nieznanego obiektu $d \in D$ do poszczególnych klas D_i indeksowanych numerami $i \in I$. Klas jest (z definicji) L, dlatego w wyniku odwzorowania C powstaje L liczb rzeczywistych i z tego powodu docelowym zbiorem w odwzorowaniu C jest $\mathcal{R}^L$.

Realizacja odwzorowania C jest – w ujęciu ogólnym – stosunkowo prosta. Na podstawie określonego wektora cech $\underline{x}$ obliczane są *funkcje przynależności* $C^i(\underline{x})$, $i = 1, 2, \ldots, L$. Wartości tych funkcji (których jest oczywiście L) określają miarę przynależności nieznanego obiektu d (dla którego odwzorowanie B określiło wektor cech $\underline{x}$) do poszczególnych klas D^i ($i = 1, 2, \ldots, L$). Odnośnie funkcji przynależności możemy jedynie sformułować postulat, aby wszystkie konkretne obiekty d^k należące do określonej klasy o numerze i^k były *wskazywane* maksymalną wartością funkcji przynależności $C^{i^k}(\underline{x}^k)$, która powinna być większa od wszystkich pozostałych *wartości* $C^i(\underline{x}^k)$ dla $i \neq i^k$.

Oczywiście postulat taki znacznie łatwiej sformułować, niż go praktycznie spełnić. Dlatego musimy rozważać dalej *wiele* różnych metod rozpoznawania, gdyż w literaturze proponuje się rozmaite definicje funkcji

$C^i(\underline{x})$ i diametralnie różniące się techniki ich konstruowania. Trzeba bowiem stwierdzić z naciskiem, że poszczególne rozważane dalej metody rozpoznawania różnią się *głównie* sposobem realizacji odwzorowania C, przeto będzie ono stale w centrum naszego zainteresowania podczas wszystkich dalszych rozważań.

W związku z wprowadzoną na końcu poprzedniego podrozdziału sugestią, że możliwe i celowe jest niekiedy podejmowanie próby rozpoznawania opartej na niepełnej znajomości wektora cech $\underline{x}$ – celowe jest tu także zasygnalizowanie korzyści, jakie odnieść można dzięki stosowaniu uogólnionego odwzorowania C. To uogólnione odwzorowanie także oznaczymy wyróżnikiem e i zdefiniujemy poszerzając odpowiednio dziedzinę we wzorze (9):

$$C^e : 2^X \to \mathcal{R}^L.$$

Stopień ogólności odwzorowania zadanego tym wzorem bywa niekiedy trudny do praktycznej realizacji, gdyż stosunkowo trudno jest zbudować funkcję, która może być równie łatwo obliczana dla dowolnej liczby m argumentów ($1 < m < n$) i w dodatku musi spełniać wymienione postulaty różnicowania obiektów należących do różnych klas. Niemniej w rozdziale 8 przedstawiona zostanie jedna z możliwych funkcji tego typu, nadająca się do tego, by praktycznie relizować postulat rozpoznawania etapowego.

2.6. Podejmowanie decyzji

Obecnie przystąpimy do dyskusji odwzorowania F. Odwzorowanie to, jak wynika z jego ogólnego zapisu:

$$F : \mathcal{R}^L \to I \cup \{i_o\}$$

lub

$$F : \mathcal{R}^L \to 2^I$$

służy do ustalenia ostatecznej decyzji (definitywnego rozpoznania – lub jego braku). W świetle wymagań, jakie sformułowano w odniesieniu do funkcji przynależności $C^i(\underline{x})$, zasada podejmowania wspomnianej decyzji jest dość prosta i oczywista. Zazwyczaj przyjmuje się *regułę majoryzacyjną*, opisaną następującym wzorem:

$$\forall_{\underline{x}\in X}\left[\left[F(C^1(\underline{x}), C^2(\underline{x}), \ldots, C^L(\underline{x})) = i\right] \equiv \forall_{\substack{\eta\in I \\ \eta\neq i}}\left[C^\eta(\underline{x}) < C^i(\underline{x})\right]\right]. \quad (12)$$

Innymi słowy, podejmowana jest decyzja o przynależności obiektu $d \in D$ opisywanego wektorem cech $\underline{x} \in D$ do tej klasy $i \in I$, dla której wartość funkcji przynależności $C^i(\underline{x})$ jest maksymalna. Decyzję neutralną i_o można przy tym podejmować w następujących przypadkach (do wyboru przez projektanta algorytmu):

1. Gdy stopień dominacji funkcji przynależności $C^\mu(\underline{x})$ maksymalnej (co do bieżącej wartości w punkcie $\underline{x}$) nad kolejną następną co do wartości funkcją przynależności $C^\nu(\underline{x})$ jest zbyt mały (mniejszy od założonej wartości ε)

$$\forall_{\substack{\eta \in I \\ \eta \notin \mu}} \left[C^\eta(\underline{x}) \leq C^\mu(\underline{x})\right] \wedge \exists_{\nu \in I} \left[C^\mu(\underline{x}) - C^\eta(\underline{x}) < \varepsilon\right]. \tag{13}$$

2. Gdy wartość dominującej funkcji przynależności $C^\mu(\underline{x})$ jest za mała (mniejsza od założonej wartości progowej ε)

$$\forall_{\substack{\eta \in I \\ \eta \notin \mu}} \left[C^\eta(\underline{x}) < C^\mu(\underline{x})\right] \wedge C^\mu(\underline{x}) < \varepsilon. \tag{14}$$

3. Gdy stosunek wartości dominującej funkcji przynależności $C^m(x)$ do sumy wszystkich wartości funkcji przynależności $C^\nu(x)$ nie wskazuje na to, że dominacja ma charakter zdecydowany i jednoznaczny

$$\forall_{\substack{\eta \in I \\ \eta \notin \mu}} \left[C^\eta(\underline{x}) < C^\mu(\underline{x})\right] \wedge \frac{C^\mu(\underline{x})}{\sum_{\nu=1}^{L} C^\nu(\underline{x})} < \frac{1}{1 + L\varepsilon}. \tag{15}$$

W przypadkach zdefiniowanych wzorami (13), (14), (15), rozpoznanie klasy μ uważamy za słabo udokumentowane (wzory (14) i (15)) lub niejednoznaczne (wzór (13)) i powstrzymujemy się od podjęcia decyzji $\hat{A}(d) = \mu$, przyjmując $\hat{A}(d) = i_o$. Oczywiście możliwe jest i inne rozstrzygnięcie, na przykład przy założeniu dopuszczalności rozpoznania $\hat{A}(d)$ w postaci podzbioru $I^d \in 2^I$ możemy przyjąć zasadę, że wskazane będą wszystkie te klasy $\mu \in I$, dla których funkcje $C^\mu(\underline{x})$ są dostatecznie duże (w praktyce – większe od pewnej ustalonej wartości ε)

$$I^d = \{\mu,\ \mu \in I \wedge C^\mu(\underline{x}) > \varepsilon\}. \tag{12a}$$

W praktyce jednak rzadko akceptujemy definicję odwzorowania F dopuszczającą rozpoznania wieloznaczne, a ponadto dla konkretyzacji dalszych rozważań na coś trzeba się jednoznacznie zdecydować. Dlatego jedynie definicja dana wzorem (12) będzie w dalszym ciągu uważana za obowiązującą i nie będzie podlegała dyskusji, natomiast konstrukcja odwzorowania C będzie tak wybierana, aby decyzje podejmowane na podstawie odwzorowania F zapewniały minimalizację funkcjonału $Q(A, \hat{A})$.

Przed ostatecznym zamknięciem tego tematu warto jeszcze wspomnieć o podejściu do zadania podejmowania decyzji przy rozpoznawaniu etapowym. Jak wspomniano, odwzorowania B oraz C mają wersje B^e i C^e, pozwalające na próbę podjęcia decyzji w warunkach braku wartości niektórych cech $x_j \in \underline{x}$, w wyniku czego $\underline{x} \in 2^X$, a nie po prostu $\underline{x} \in X$. Domknięciem tych odwzorowań jest F^e zdefiniowane jako

$$F^e : \mathcal{R}^L \to I \cup \{i_o, i_e\},$$

gdzie nowy symbol i_e oznacza decyzję: *brak możliwości rozpoznania, należy zmierzyć kolejne cechy i ponowić próbę rozpoznawania.* Decyzja i_e w warunkach, kiedy pomierzono już wszystkie pozostające do dyspozycji cech, staje się równoważna decyzji i_o, lecz ta ostatnia może być także podjęta w sposób kategoryczny jeszcze przed pomierzeniem wszystkich cech $x_j \in \underline{x}$.

2.7. Ciąg uczący

Twórca algorytmu $\hat{A}$ dysponuje zazwyczaj jedynie wiedzą na temat ciągu uczącego U, nie ma natomiast dostępu do informacji charakteryzujących w całości odwzorowanie A. W związku z tym dalsze rozważania na temat konstrukcji funkcji przynależności tworzących odwzorowanie C są prowadzone w ten sposób, aby bazowały na wykorzystaniu ciągu uczącego U. Ciąg ten może być wykorzystywany automatycznie, jak w metodach opisanych w rozdziałach 4, 5, 6 i 7, może wymagać oddzielnego opracowania (na przykład statystycznego – rozdz. 8), ewentualnie może być generowany przez założone reguły (rozdz. 9, 10, 11 i 12). W każdym z wymienionych przypadków ciąg U można zdefiniować jako zbiór par

$$U = \{(\underline{x}^k, i^k), \quad k = 1, 2, \ldots, N\}, \tag{16}$$

gdzie

$$\underline{x}^k = B(d^k) \wedge d^k \in D \tag{17}$$

oraz

$$i^k \in I \wedge i^k = A(d^k). \tag{18}$$

Elementy ze zbioru U będziemy nazywać *przykładami*. Jak wynika z prztoczonych zapisów, każdy przykład składa się z pełnej charakterystyki (kompletnego wektora cech $\underline{x}^k$) pewnego obiektu d^k oraz z informacji na temat numeru klasy i^k, do której obiekt ten powinien być zaliczony.

Dla wygody dalszych rozważań zbiór U można zdekomponować na L podzbiorów U^i $(i = 1, 2, \ldots, L)$ w ten sposób, aby w i-tym podzbiorze znajdowały się wyłącznie obiekty należące do i-tej klasy

$$U = \bigcup_{i \in I} U^i, \tag{19}$$

$$U^i = \{\underline{x}^{i,k}\}, \quad k = 1, 2, \ldots, N^i, \tag{20}$$

$$\forall_{i \in I} \left[x^{i,\nu} \in U^i \equiv \langle \underline{x}^\mu, i^\mu \rangle \in U \wedge \underline{x}^\mu = \underline{x}^{i,\nu} \wedge i^\mu = i\right]. \tag{21}$$

Wybór elementów d^k należących do ciągu uczącego U powinien zapewniać jego *reprezentatywność*. W praktyce ciąg ten zazwyczaj stanowi próbkę losowo pobraną ze zbioru D, przeto jego reprezentatywność może być dyskusyjna. Jedną z metod polepszania reprezentatywności ciągu uczącego jest jego wydłużanie (zwiększanie N).

3. KLASYFIKACJA METOD ROZPOZNAWANIA(1)

3.1. Potrzeba klasyfikacji metod rozpoznawania obrazów

Obfitość literatury z zakresu rozpoznawania obrazów(2) powoduje, że liczba różnych metod rozpoznawania sięga setek. W dodatku niemal wszystkie metody wprowadzane są i omawiane przez swoich autorów w oderwaniu od pozostałych, opisywanych przez innych badaczy. W tej sytuacji, podejmując próbę zbiorczego opisania różnych metod w jednym podręczniku trzeba (obok uzgodnień notacji matematycznej i terminologii) dokonać próby usystematyzowania zbioru rozważanych metod. Punktem wyjścia do takiej systematyzacji może być próba klasyfikacji opisywanych w literaturze metod. Klasyfikacji takiej nie dokonał żaden z autorów znanych monografii czy zbiorczych opracowań, co powoduje, że przytoczona próba stanowi prezentację subiektywnego punktu widzenia i może być przedmiotem krytyki [57].

---

(1) Rozdział ten może być pominięty przy pierwszym czytaniu.

(2) Ilościowe oszacowanie tej literatury jest dość trudne, jednak pewne jest, że są to tysiące. Same tylko bibliografie podane w monografiach [11], [12] lub [13] zawierają łącznie około 1,5 tysiąca prac, a można twierdzić, że nie są to bibliografie kompletne. W dodatku stale pojawiają się nowe prace, dotyczące tej dziedziny. Wydawane są nawet całe periodyki, poświęcone tylko tej problematyce (Pattern Recognition, Computer Vision czy słynne PAMI - IEEE Transactions on Pattern Analysis and Machine Intelligence), a w wielu innych pismach naukowych stale publikuje się coraz to nowe artykuły poświęcone rozpoznawaniu (w Polsce: Archiwum Automatyki i Telemechaniki, Podstawy Sterowania, Postępy Cybernetyki, Elektrotechnika i wiele innych).

3.2. Zasada podziału i podstawa klasyfikacji metod rozpoznawania

Podziału metod rozpoznawania dokonamy na podstawie klasyfikacji trzech wprowadzonych odwzorowań: B, C oraz F, składających się na rozpoznawanie $\hat{A}$. Do pełnego scharakteryzowania rozważanej metody rozpoznawania wystarczyć może podanie wyróżników tych odwzorowań, wybranych według pewnego ustalonego kodu. Proponowana tu forma kodu ma postać:

$$F_\eta C_\nu B_\mu,$$

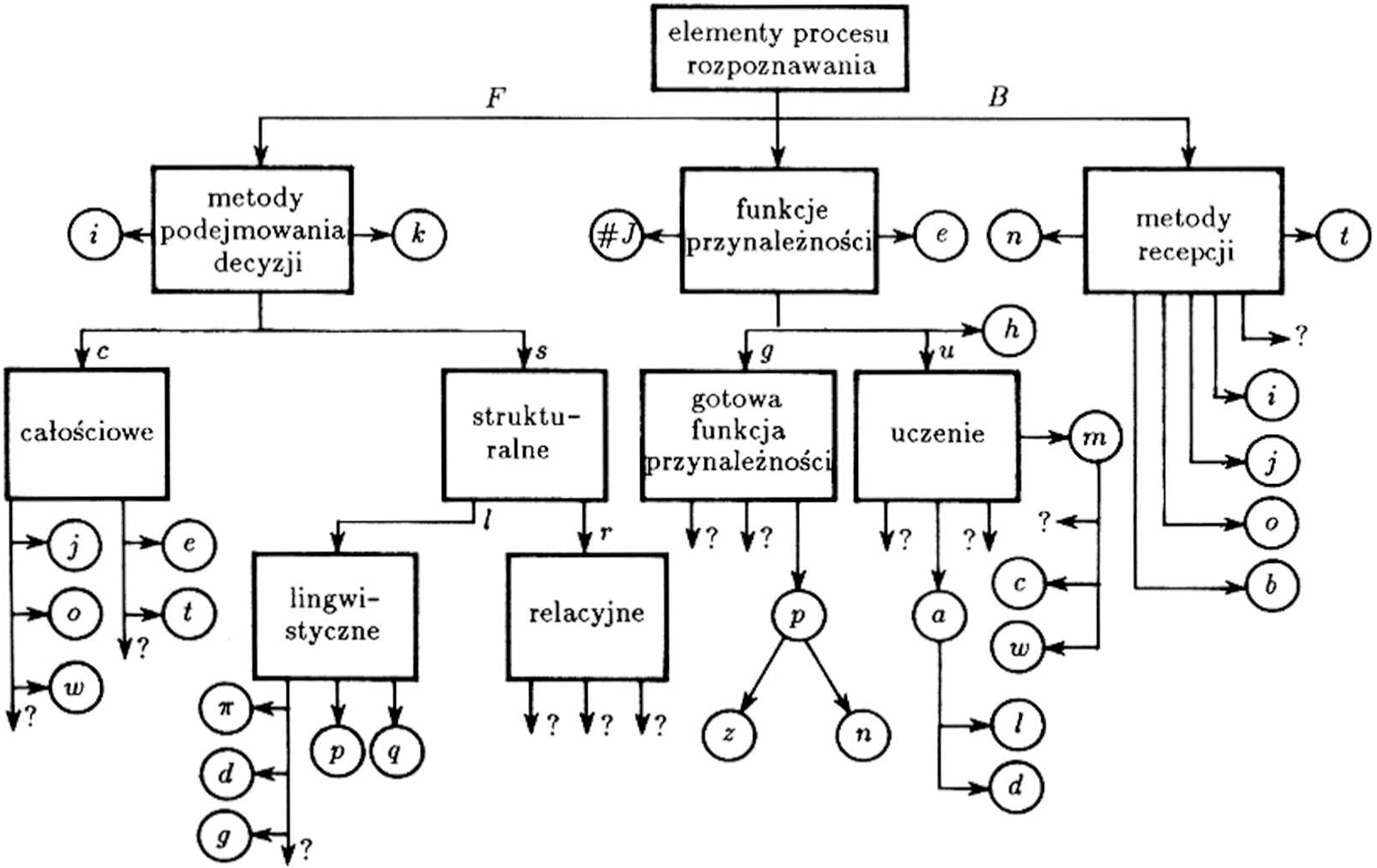
gdzie μ, ν oraz η są pewnymi wyróżnionymi identyfikatorami, wskazującymi jednoznacznie na rodzaj używanej reguły podejmowania decyzji, funkcji przynależności i funkcji recepcji. Dla prostoty i mnemoniczności powstających w ten sposób charakterystyk rozważanych metod rozpoznawania $\hat{A}$ będziemy dążyli do tego, aby identyfikatory μ, ν lub η stanowiły jednoliterowe skróty odpowiednich słów, opisujących atrybuty poszczególnych etapów procesu rozpoznawania (rys. 3.1).

Kategorie wprowadzone dla odwzorowań F, C i B nie są wzajemnie wykluczające ani nie są rozłączne, zatem przy opisie konkretnej metody rozpoznawania może okazać się potrzebne wprowadzenie *kilku* identyfikatorów μ, ν lub η

$$F_{\nu_1\nu_2\nu_3} C_{\nu_1\nu_2} B_{\mu_1\mu_2\mu_3\mu_4}.$$

Taki zapis dla większości zadań rozpoznawania pozwala określić używane metody. W celu zapewnienia jednoznaczności unikać należy powtarzania się tych samych identyfikatorów μ, ν lub η, denotujących różne kategorie dla *tego samego odwzorowania*([3]) rozważanej metody rozpoznawania. Będzie to niekiedy wymagało zaakceptowania pewnych nowych konwencji

([3]) Unikać będziemy więc powtarzających się symboli η przypisanych różnym kategoriom F (reguł podejmowania decyzji), wyeliminujemy powtarzające się symbole ν odpowiadające kategoriom C (funkcji przynależności) i zabezpieczymy się przed powtórzeniami w zakresie symboli μ. Natomiast za dopuszczalne uważać będziemy takie same symbole przyjmowane dla – przykładowo – ν i η, chociaż oczywiście oznaczać one będą zupełnie różne rzeczy (ν odpowiada rodzajowi funkcji przynależności, a η regule podejmowania decyzji). Jednak sama forma zapisu identyfikatorów jako indeksów symboli F, C i B zapobiega w tym przypadku niejednoznaczności, a wyszukiwanie różniących się symboli dla wszystkich możliwych elementów klasyfikacji okazało się zbyt uciążliwe.



Rys. 3.1. Graf współzależności między kryteriami klasyfikacji

w zakresie terminologii, bowiem tradycyjne terminy w wielu przypadkach zaczynają się od identycznych liter, a będziemy unikali stosowania skrótów czysto umownych. W przypadku używania metod *hybrydowych*, to znaczy rozwiązywania jednego zadania przy użyciu *kilku* metod, stosować można zapis (wzorowany na klasyfikacji UKD) wykorzystujący dwukropek jako separator alternatywnych charakterystyk, na przykład

$$F_{\nu_1\nu_2} : F_{\nu_3}C_{\nu_1} : C_{\nu_2} : B_{\mu_1} : B_{\mu_2} : B_{\mu_3\mu_4}.$$

Jak wspomniano, identyfikatory μ, ν i η mają charakter umowny i muszą być wybierane z pewnej listy, którą spróbujemy wstępnie zaproponować, ale traktować ją trzeba oczywiście jako otwartą, zarówno z powodu ograniczonego zakresu i niekompletności prezentowanych w książce algorytmów i metod, jak ze względu na powstające wciąż nowe techniki rozpoznawania.

3.3. Klasyfikacja metod podejmowania decyzji

Zaczniemy od kryteriów klasyfikacji najbardziej ogólnej, związanej z metodami podejmowania decyzji F, a w istocie – z samą podstawową „filozofią" rozważanych metod rozpoznawania. Możliwości jest tu bardzo wiele, jednak nie wdając się w niuanse możemy generalnie wydzielić podejście *całościowe* oraz *strukturalne*. Odpowiednie identyfikatory mogą być przyjęte w postaci

$\eta = c$ dla metod całościowych,

$\eta = s$ dla metod strukturalnych.

W obrębie tych generalnych klas można wydzielić pewne ważne podklasy, wskazywane – w razie potrzeby – poprzez uzupełnienie identyfikatora c lub s dodatkowymi symbolami. Metody całościowe ($\eta = c$) można dalej różnicować na dostarczające jednoznacznych rozwiązań ($F : \mathcal{R}^L \to I$) oraz takie, które mogą odmawiać odpowiedzi ($F : \mathcal{R}^L \to I \cup \{i_o\}$), albo nawet dawać rozpoznania wieloznaczne ($F : \mathcal{R}^L \to 2^I$). Odpowiednie symbole można zaproponować w postaci

$\eta = j$ dla metod jednoznacznych,

$\eta = o$ dla metod przewidujących odmowę rozpoznania,

$\eta = w$ dla metod rozpoznań wielowariantowych.

Możliwe jest także wprowadzenie dalszych podziałów i dalszych identyfikatorów. Metody całościowe ($\eta = c$) na ogół preferują technikę rozpoznawania polegającą na *jednorazowym* podejmowaniu decyzji. Istnieje jednak podgrupa tych metod ($\eta = c$), której charakterystyczną cechą jest możliwość podejmowania decyzji na podstawie znajomości tylko *niektórych* cech tworzących przestrzeń X. Metody te nazwiemy *etapowymi* i oznaczymy identyfikatorem $\eta = e$, zaś typowe metody, w których decyzje podejmuje się na podstawie znajomości wszystkich cech oznaczać będziemy identyfikatorem $\eta = t$ (od określenia *totalne*([4])). Mamy więc

$\eta = e$ dla metod rozpoznawania etapowego([5]),

$\eta = t$ dla rozpoznawania całościowego (totalnego).

Identyfikator $\eta = t$ można pomijać, ponieważ częstość występowania właśnie tej sytuacji skłania do uznania tego identyfikatora za standard nie wymagający jawnej deklaracji([6]).

W klasie metod strukturalnych ($\eta = s$) ważną podklasę stanowią metody bazujące na podejściu *lingwistycznym*([7]), które przeciwstawia się innym metodom strukturalnym (na przykład relacyjnym). Tak więc wprowadzając identyfikatory rozszerzające klasyfikację $\eta = s$ możemy zaproponować od razu na wstępie dwa uzupełnienia:

$\eta = l$ dla metod lingwistycznych,

$\eta = r$ dla metod relacyjnych.

Oczywiście należy oczekiwać, że zachodzący obecnie intensywny rozwój metod strukturalnych w niedługim czasie listę tę znacznie wydłuży. Metody ligwistyczne ($\eta = l$), w których rozpoznawanie obrazów jest traktowane jako problem badania struktury rozpoznawanych obiektów metodami bazującymi na gramatykach formalnych, mogą być przedmiotem dalszych podziałów ze względu na rozmaity charakter używanych gramatyk. Ograniczając rozważania do zagadnień rozpoznawania obrazów w węższym tego słowa znaczeniu (to znaczy koncentrując się na strukturalnej analizie dwu- i trójwymiarowych złożonych scen), możemy zaproponować wykorzystanie

([4]) Bardziej naturalna nazwa „metody całościowe" nie może być użyta ze względu na jednoznaczność identyfikatorów.

([5]) Częściej stosowana nazwa „metody sekwencyjne" nie może być użyta ze względu na jednoznaczność identyfikatorów.

([6]) Na podobnej zasadzie funkcjonuje tzw. *default* w informatyce.

([7]) W literaturze metody te znane są także pod nazwą metod *syntaktycznych*.

gramatyk *łańcuchowych* (ciągowych), *drzewowych* lub *grafowych*. Pozwala to na wprowadzenie kolejnych trzech identyfikatorów:

$\eta = \acute{l}$ dla metod gramatyk łańcuchowych,
$\eta = d$ dla metod gramatyk drzewowych,
$\eta = g$ dla metod gramatyk grafowych.

Specyficznym podejściem strukturalnego (lingwistycznego) typu jest stosowana w zadaniach rozpoznawania mowy metoda oparta na zasadzie *programowania dynamicznego*. Bliższe szczegóły na temat tej metody podano w zakończeniu części szczegółowej książki. Jako alternatywę dla metod programowania dynamicznego w zadaniach rozpoznawania mowy można rekomendować metody oparte na analizie kontekstu rozpoznawanej części zdania z wykorzystaniem *sensu* całej wypowiedzi. Ponieważ automatyzacja analizy znaczeniowej języka naturalnego jest chwilowo kwestią dość odległej przyszłości, przeto odpowiednie metody nazwiemy *quasi-semantycznymi* i możemy bez większego ryzyka przewidzieć, że będą się one rozwijać i zyskiwać na znaczeniu, co zapewne spowoduje konieczność zastosowania w przyszłości bardziej precyzyjnych kwalifikacji. Na teraz wystarczy jednak wprowadzić identyfikatory $\eta = p$ oraz $\eta = q$ dla tej kategorii metod

$\eta = p$ dla metod programowania dynamicznego,
$\eta = q$ dla metod quasi-semantycznych.

Metody bazujące na podejściu lingwistycznym ($\eta = l$) są aktualnie tak silnie rozwijane, a jednocześnie charakteryzują się tak dużą liczbą cech specyficznych, odmiennych od pozostałych metod rozpoznawania, że powinny być rozpatrywane *oddzielnie* od wszystkich innych metod strukturalnych ($\eta = s$) i całościowych ($\eta = c$) – co znajduje odbicie w dalszej treści książki.

Tradycyjnie w literaturze dotyczącej rozpoznawania obrazów zamieszcza się także opisy metod służących do realizacji odwzorowania *odwrotnego* do rozpoznawania, a mianowicie automatycznego grupowania obiektów w klasy. Metody te nie są (jak już wzmiankowano) omawiane w tej książce, jednak uwzględniając fakt, że podstawowa odmienność metod grupujących polega na odmiennym charakterze podejmowanych przez te metody *decyzji*, celowe wydaje się wspomnienie o nich w ramach klasyfikacji przy okazji atrybutów nadawanych charakterystyce odwzorowania F. Wzmiankowane metody są rozmaicie nazywane w literaturze: klasteringiem (lub klasteryzacją – od angielskiego terminu *cluster analysis)*, uczeniem bez nauczyciela (lub samouczeniem), taksonomią itp. W książce przyjęto generalnie nazwę metod *grupujących*, jednak dla jednoznaczności iden-

tyfikatora warto określić te metody jako *klasyfikcję* nadając im identyfikator $\eta = k$, w odróżnieniu od metod typowego rozpoznawania, które można określić jako *identyfikację* i związać z identyfikatorem([8])

$\eta = i$ dla klasycznego rozpoznawania (identyfikacji),

$\eta = k$ dla metod klasyfikacji (klasteryzacji).

3.4. Podział funkcji przynależności

Najwięcej ciekawych możliwości podziału i klasyfikacji rozważanych metod rozpoznawania wynika ze sposobu realizacji odwzorowania C, rozpatrywanego przy założeniu, że odwzorowanie F jest zadane w jeden z omówionych sposobów. W omawianym systemie klasyfikacji, stosowne identyfikatory ν, wynikające z sugerowanych podziałów klasyfikacyjnych, przypisywane są symbolowi C w każdym konkretnym zapisie wprowadzonego umownego kodu.

Główna charakterystyka, jaką można wprowadzić dla opisania stopnia złożoności odwzorowania C, wiąże się z liczbą rozpoznawanych klas L. Jest oczywiste, że inny jest stopień komplikacji dla zadania dychotomizacji ($L = 2$), inny zaś dla zadania rozpoznawania jednej z kilkudziesięciu klas. Dlatego na początku charakterystyki zadanej identyfikatorami ν dla odwzorowania C podaje się *liczbę* L określającą moc zbioru I

$\nu = \#I$ liczba rozpoznawanych klas.

Odpowiedni zapis ma postać C_2 dla zadania rozdziału dwóch klas i C_{40} dla rozpoznawania jednego z około czterdziestu zwyczajowo wyróżnianych fonemów (głosek) języka polskiego.

Podstawowy podział, jaki można wprowadzić dla funkcji przynależności dotyczy sposobu *tworzenia* tych funkcji, a to sprowadza się do zagadnienia obecności lub braku w rozważanej metodzie elementu *uczenia*. Jako pierwszy wprowadzimy więc identyfikator $\nu = u$ wyróżniający te metody, które przy budowie funkcji przynależności uwzględniają proces uczenia. Przeciwstawimy je metodom z *gotową* funkcją przynależności, w których proces uczenia zastąpiony jest pewnym apriorycznym rozumowaniem twórcy metody. Czasem zresztą rozumowanie to zredukowane bywa do kilku pomysłów czysto *heurystycznej* natury, co szczególnie często ma miejsce w meto-

([8]) Oczywiście także i ten identyfikator można pomijać jako *default*.

dach rozpoznawania przeznaczonych do konkretnych zastosowań (na przykład rozpoznawanie liter([9]))

$\nu = u$ dla metod z uczeniem,
$\nu = g$ dla metod z gotową funkcją przynależności,
$\nu = h$ dla metod heurystycznych.

Metody nie zawierające elementu uczenia są w książce traktowane w sposób skrótowy, zaś ich dalsza klasyfikacja (związana z wprowadzaniem nowych identyfikatorów) musi być odłożona do momentu bardziej istotnego rozwinięcia i usystematyzowania tych metod, które obecnie służą raczej do zaspokajania doraźnych potrzeb praktyki, a nie rozwojowi ogólnej teorii. W dalszych rozdziałach książki metody nie zawierające pierwiastka uczenia nie są praktycznie omawiane.

Stosowane niekiedy w literaturze dalsze podziały metod zawierających uczenie, na przykład wyróżniające *uczenie z nauczycielem* i *uczenie bez nauczyciela* lub *uczenie parametryczne* i *uczenie nieparametryczne* – uwzględniono w postaci odmiennego zaklasyfikowania w zakresie innych identyfikatorów (na przykład uczenie *bez nauczyciela* identyfikowane jest przez $\eta = k$, a uczenie *parametryczne* – za pomocą $\nu = p$).

Inna, równie ogólna klasyfikacja wykorzystująca odwzorowanie C pozwala wydzielić metody, w których funkcje przynależności budowane są od razu dla *wszystkich* elementów ciągu uczącego, albo są budowane w ten sposób, aby mogły być obliczane na podstawie niektórych tylko *składowych*.

Funkcje przynależności mające własność denotowaną jako $\nu = s$ mogą być wykorzystywane przy etapowych metodach podejmowania decyzji ($\eta = e$), chociaż nie jest to konieczne, gdyż sensowne może być wykorzystanie funkcji tej klasy przy próbach podejmowania decyzji jednorazowych ($\eta = i$) – na przykład w sytuacjach kiedy część cech tworzących wektor $\underline{x}$ jest całkowicie nieosiągalna – bez możliwości uzupełnienia brakujących danych (na

---

([9]) Technika automatycznego odczytywania tekstów drukowanych (a niekiedy nawet starannych rękopisów) ma ogromne znaczenie praktyczne (wprowadzanie treści różnych dokumentów do systemów komputerowych, automatyczne odczytywanie adresów przesyłek itp.). z tego powodu buduje się i sprzedaje na świecie bardzo wiele urządzeń określanych jako skanery (od angielskiego *scaner)* lub nazywanych krótko OCR (*optical character reader*). Urządzenia te z reguły oparte są na pomysłowych „sztuczkach" i niewiele mają wspólnego z teorią rozpoznawania obrazów.

przykład rozpoznawanie odcisku palca, który zdjęto z przedmiotu zbrodni w postaci drobnego fragmentu rysunku linii daktyloskopijnych).

Dalsze elementy klasyfikacji i dalsze identyfikatory ν uzyskamy rozważając bliżej samą metodę konstruowania funkcji przynależności. W najprostszym ujęciu podejście do konstrukcji odwzorowania C może bazować na rozważaniach natury geometrycznej, prowadzonych w przestrzeni X i opartych na takich pojęciach, jak *sąsiedztwo* lub *odległość*. Odpowiednią grupę metod nazwiemy generalnie metodami *minimalnoodległościowymi* i związemy z identyfikatorem $\nu = m$

$\nu = m$ metoda minimalnoodległościowa.

W obrębie metod minimalnoodległościowych wyróżnić można dwie kategorie: większość tych metod wykorzystuje ciąg uczący w *całości*, bez jakiegokolwiek głębszego przetwarzania. Część metod proponuje jednak przetworzenie ciągu uczącego, niekiedy nawet bardzo wyrafinowane, do postaci odpowiednio definiowanych *wzorców*. Wprowadzimy więc dodatkowe identyfikatory $\nu = w$ oraz $\nu = c$:

$\nu = c$ metoda wykorzystuje cały ciąg uczący,

$\nu = w$ metoda wykorzystuje wzorce.

W innych metodach można konstruować odwzorowanie C na podstawie metod zbliżonych do metod aproksymacji funkcji. Odpowiednią grupę metod nazwiemy metodami *aproksymacyjnymi*. Wprowadzimy tu identyfikator $\nu = a$

$\nu = a$ metoda aproksymacyjna.

Metod aproksymacyjnych jest wiele i można je dzielić według różnych kryteriów. W literaturze funkcjonują rozmaite nazwy (na przykład „metoda funkcji potencjalnych"), które dzielą te metody raczej ze względu na to, *kto je opracował*, a nie *jakie mają właściwości*. Z punktu widzenia charakterystyki samych metod można dokonać bardziej ogólnych podziałów, na przykład celowe jest wydzielenie – jako specyficznej podklasy – zbioru metod, w których funkcje C^i są *liniowe*, oraz zbioru (obszerniejszego, ale mniej użytecznego praktycznie) metod, w których funkcje C^i mogą być *dowolne*:

$\nu = l$ metoda funkcji liniowych,

$\nu = d$ metoda wykorzystująca dowolne funkcje.

Pozostałe podklasy metod aproksymacyjnych mogą być wprowadzane w miarę potrzeby, jednak ze względu na obfitość różnorodnych propozycji

spotykanych w literaturze trzeba tu zachować umiar i wprowadzać dalsze symbole klasyfikacyjne jedynie wtedy, gdy rzeczywiście są one potrzebne (ponieważ wydzielają klasy istotnie nowe) z punktu widzenia istoty metody rozpoznawania.

Funkcje przynależności budujące odwzorowanie C można także otrzymać w wyniku rozważań prowadzonych na gruncie rachunku prawdopodobieństwa. Odpowiednią grupę metod nazwiemy metodami *probabilistycznymi*, wprowadzając kolejny identyfikator:

$\nu = p$ metoda probabilistyczna.

W metodach tej grupy także możliwa jest dalsza klasyfikacja, przyjmująca za podstawę rozmaite kryteria. Za najważniejszy uznać można podział wynikający z założeń dotyczących charakteru wchodzących w rachubę rozkładów prawdopodobieństwa. Możliwe są dwie sytuacje: albo zakładamy, że rozkład jest *znany*, albo nie:

$\nu = z$ znany rozkład prawdopodobieństwa,

$\nu = n$ nieznany rozkład prawdopodobieństwa.

Sytuacja, kiedy rozkład jest znany jest na ogół łatwiejsza, gdyż zadanie uczenia (jeśli się takie wprowadza) polega w tym przypadku na estymacji parametrów([10]) tego rozkładu. Przy braku sensownych założeń odnośnie charakteru rozkładu zadanie rozpoznawania staje się trudniejsze, a uczenie jest bardziej pracochłonne. Jednak w takim przypadku unika się przynajmniej części kłopotów wywoływanych mało trafnymi założeniami odnoszącymi się do charakteru rozkładu.

3.5. Podział metod recepcji

Klasyfikacja oparta na odwzorowaniu B jest najmniej wdzięczna, ponieważ recepcja jest zawsze silnie uzależniona od właściwości rozpoznawanych obiektów $d \in D$, a więc od uwarunkowań słabo zależnych od właściwości budowanej metody rozpoznawania, a zasadniczo zależnych od fizycznej natury rozważanego obszaru zastosowań. Jednak kilka przynajmniej propozycji wartości identyfikatorów μ można wstępnie zaproponować, wyrażając nadzieję, że postęp badań nad systematyzacją metod rozpoznawania przyniesie także i w tym zakresie bardziej konstruktywne rozwiązania.

([10]) Na przykład średnich i wariancji.

Podobnie jak w odniesieniu do identyfikatorów ν, zastosujemy na początku charakterystykę określającą rozmiar zadania rozpoznawania. Wyróżnikiem będzie więc rozmiar przestrzeni cech n:

$\mu = n$ liczba określająca wymiar przestrzeni cech.

Podstawowy podział metod recepcji B może być dokonany w zależności od tego, jakiego rodzaju cechy $x_j \in x$ są w niej wyróżniane. Możliwe jest w szczególności wyróżnienie odwzorowań dostarczających cech *ilościowych* (możliwych do rozważania dalej jako konkretne liczby), cech *jakościowych* (możliwych do wyrażenia za pomocą kodów ustalających wartości cech w skali porządkowej, co pozwala na prowadzenie porównań typu *większy* i *mniejszy*, ale bez wartościowania), cech *opisowych* (pozwalających na rozróżnianie pewnych sytuacji i na wykrywanie identyczności pewnych cech, lecz nie pozwalających na ich wzajemne porównywanie w jakiejkolwiek skali odniesień względnych), wreszcie cech *binarnych* (które można tylko rozpatrywać w kategoriach obecności lub braku pewnych właściwości). Możliwe jest więc wprowadzenie identyfikatorów:

$\mu = i$ dla metod dostarczających cech ilościowych,
$\mu = j$ dla metod dostarczających cech jakościowych,
$\mu = o$ dla metod dostarczających cech opisowych,
$\mu = b$ dla metod dostarczających cech binarnych.

Wymienione identyfikatory należą do tych, które szczególnie często muszą być używane w formie alternatywnych charakterystyk, ponieważ w wielu zastosowaniach podejmując decyzję trzeba opierać się na cechach należących – w myśl podanej klasyfikacji – do różnych klas. Na przykład w elementarnym przykładzie diagnostyki medycznej brać można pod uwagę temperaturę ($\mu = i$), ogólny stan pacjenta (w skali *dobry, zły, bardzo zły, fatalny* itp.)($\mu = j$), jego płeć ($\mu = o$) oraz występowanie lub brak wymiotów ($\mu = b$).

Jak wspomniano na wstępie, cechy, na podstawie których dokonuje się rozpoznawania mogą być rozważane wprost, albo mogą być poddawane pewnej *transformacji*. Obecność tej transformacji sygnalizować będziemy identyfikatorem

$\mu = t$ metoda wykorzystująca transformację cech.

Dalsze klasyfikacje muszą być wprowadzane sukcesywnie, według potrzeb konkretnego użytkownika.

4. METODY MINIMALNOODLEGŁOŚCIOWE

4.1. Wprowadzenie

W metodach minimalnoodległościowych odwzorowanie C wiąże się z pojęciem *odległości* w przestrzeni X, która w tym celu musi być wyposażona w odpowiednią *metrykę*. Dyskusja dokładnego sposobu zdefiniowania tej metryki, zapisywanej w sposób ogólny jako odwzorowanie:

$$\rho : X \times X \rightarrow R_* \tag{22}$$

jest przeniesiona do Dodatku 1, gdyż omawianie jej w tym miejscu zajęło by zbyt wiele miejsca. Ogólnie znane przesłanki matematyczne dają tu dużą swobodę, ponieważ metryką ρ może być dowolne odwzorowanie postaci (22), spełniające dla wszystkich wektorów $\underline{x}^\mu \in X$ $(\mu = 1, 2, \ldots)$ następujące (dość oczywiste) założenia:

$$\rho(\underline{x}^\mu, \underline{x}^\nu) = 0 \Leftarrow \underline{x}^\mu \equiv \underline{x}^\nu,$$

$$\rho(\underline{x}^\mu, \underline{x}^\nu) = \rho(\underline{x}^\nu, \underline{x}^\mu),$$

$$\rho(\underline{x}^\mu, \underline{x}^\nu) < \rho(\underline{x}^\mu, \underline{x}^\eta) + \rho(\underline{x}^\eta, \underline{x}^\nu).$$

Warto podkreślić, że takich odwzorowań, spełniających podane postulaty, jest potencjalnie nieskończenie wiele, a zatem twórca metody rozpoznawania ma w tym zakresie wiele swobody. Być może nawet nieco zbyt wiele, gdyż problem wyboru właściwej metryki można rozwiązać w zasadzie jedynie na drodze empirycznej, metodą *prób i błędów*, albo podejmując odpowiednią decyzję całkowicie arbitralnie. Równocześnie trzeba podkreślić, że przy praktycznym stosowaniu metod minimalnoodoległościowych jest to problem kluczowy, mający zasadniczy wpływ na uzyskiwane efekty.

Przykład. W badaniach nad rozpoznawaniem mowy polskiej [58] prowadzonych w latach 1970–1990 w Zakładzie Biocybernetyki AGH, przyjmowano jako składowe x_j przestrzeni cech X amplitudy sygnału dźwiękowego w 96 pasamach widma o częstotliwościach $f_j = 125j + 62{,}5$ Hz ($j = 1, 2, \ldots, 96$). W przestrzeni tej zdefiniowano 9 różnych metryk([1]), a następnie podejmowano próby rozpoznawania określonych fonemów (na przykład samogłosek) przy użyciu opisanych dalej metod z wykorzystaniem tych metryk. Okazało się, że na tym samym komputerze i dla tych samych wypowiedzi (traktowanych jako testy) procent poprawnych rozpoznań i czas rozpoznawania przedstawia się w sposób pokazany w tabeli 4.1.

Tabela 4.1. Zależność dokładności i czasu rozpoznawania od zastosowanej metryki

Metryka	Procent poprawnych rozpoznań	Czas rozpoznawania, ms
Euklidesowa	0,933	1,00
Hamminga	0,967	0,64
Czebyszewa	0,933	0,84
Camberra	0,933	0,96
Minkowskiego (0,1)	0,767	3,05
Minkowskiego (10)	0,933	2,03
Pearsona	0,900	1,03
Mahalanobisa (1)	0,833	1,30
Mahalanobisa (2)	0,833	1,30

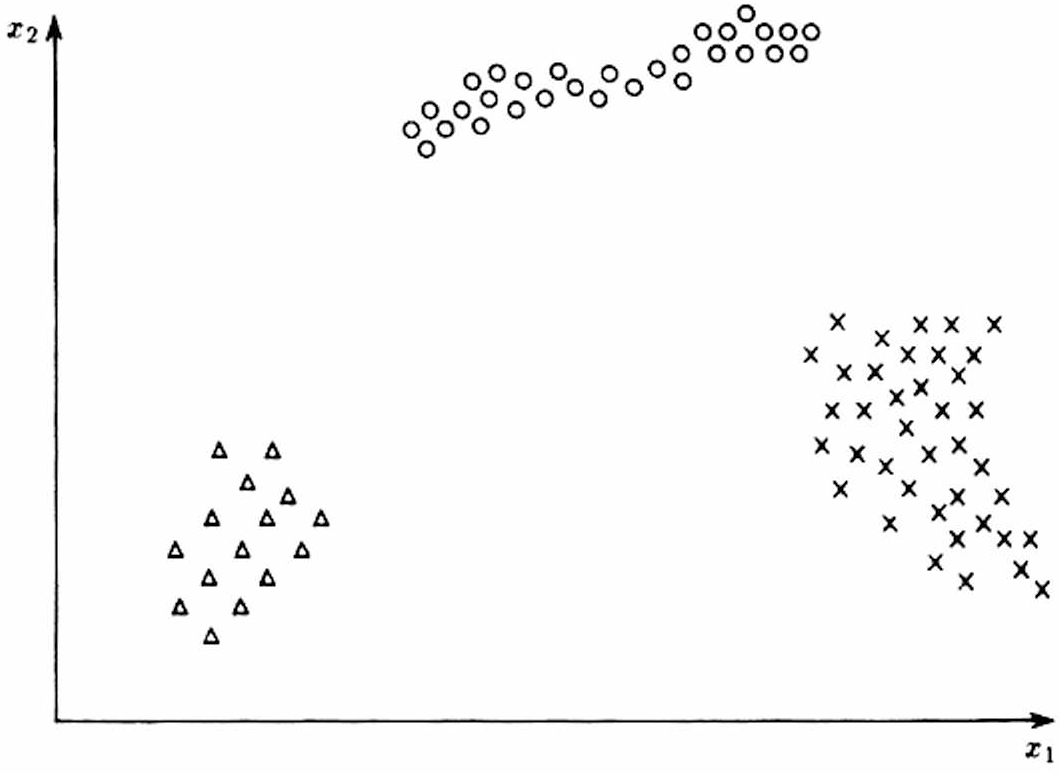
Jak wynika z tego przykładu, od trafnego wyboru metryki zależeć może zarówno efekt rozpoznawania, jak i jego pracochłonność – wyrażająca się mniejszymi lub większymi wymaganiami co do mocy obliczeniowej maszyny rozpoznającej. Rozwinięcie tego tematu także znajduje się w Dodatku 1.

4.2. Metoda *NN*

Zasada wszystkich omawianych tu metod (rys. 4.1) zwanych *minimalnoodległościowymi* polega na następującym prostym rozumowaniu: Wybrać

([1]) Niektóre z tych metryk przytoczono w Dodatku 1.

jako rozpoznanie $i \in I$ tę klasę, do której należy obiekt $x^{i,k} \in U$ najbliższy (w myśl przyjętej metryki ρ) rozpoznawanemu obiektowi d (a dokładniej reprezentującemu go wektorowi x). W najczystszej postaci algorytm ten realizuje następująca metoda.



Rys. 4.1. Metody minimalnoodległościowe są oparte na przesłankach związanych z geometrią przestrzeni cech. Jeśli punkty, odpowiadające obiektom różnych klas (na rysunku oznaczone jako trójkąty, kółka i krzyżyki) grupują się w formie wyraźnych skupisk, to wówczas możliwe i celowe jest posłużenie się pojęciem odległości przy podejmowaniu decyzji

Przyjmując, że w przestrzeni X zdefiniowano pewną metrykę, możemy odwzorowanie C zapisać w postaci:

$$C^i(\underline{x}) = \frac{1}{\rho(\underline{x}, \underline{x}^{i,k}) + \varepsilon}, \qquad i = 1, 2, \ldots, L. \tag{23}$$

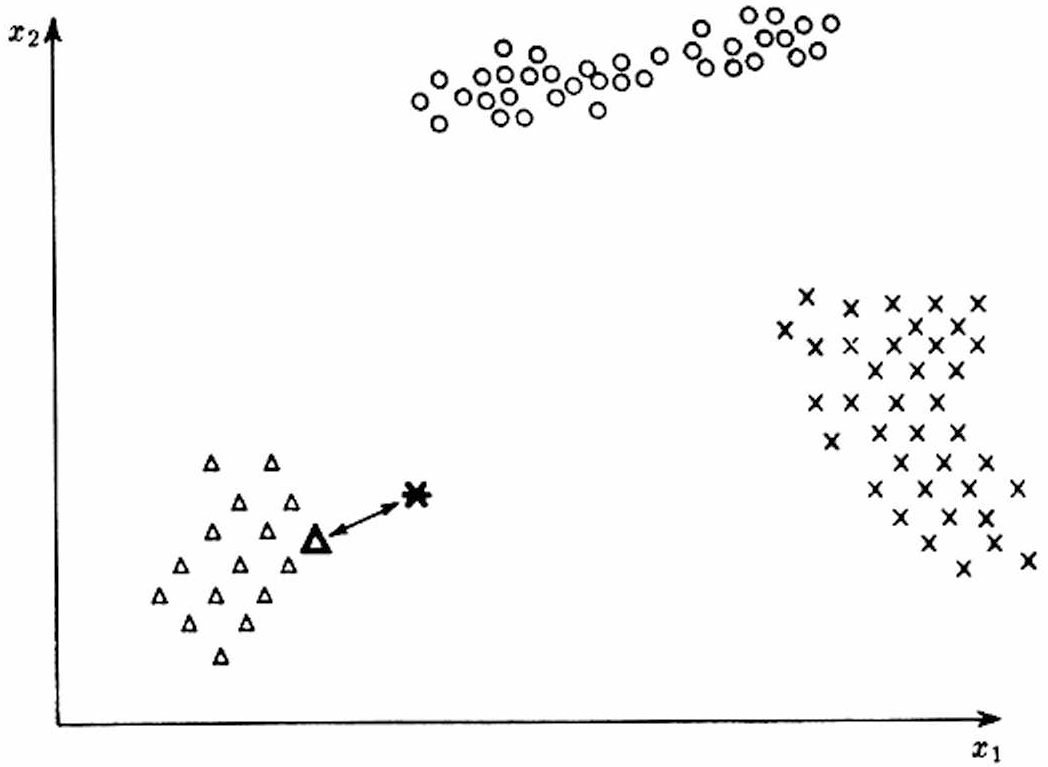
Element $\underline{x}^{i,k}$ zgodnie z przyjętymi oznaczeniami (porównaj: wzór (20)) należy do podzbioru U^i, a małą stałą dodatnią ε wprowadzono w celu zapewnienia warunku $C^i(\underline{x}) < \infty$, gdyż teoretycznie możliwe jest uzyskanie

pokrycia punktów

$$\rho(\underline{x}, \underline{x}^{i,k}) = 0.$$

W podstawowym wariancie, nazywanym algorytmem NN (od angielskich słów: *nearest neighbour - najbliższy sąsiad*), wybór elementu $\underline{x}^{i,k}$ we wzorze (23) jest dokonywany zgodnie z regułą (rys. 4.2)

$$\rho(x, x^{i,k}) = \min_{\underline{x}^{\mu} \in U'} (\underline{x}, \underline{x}^{\mu}). \tag{24}$$



Rys. 4.2. Reguła podejmowania decyzji w przypadku algorytmu NN zakłada, że nieznany obiekt (oznaczony gwiadką) zostanie zaklasyfikowany do tego obrazu, do którego należy obiekt ciągu uczącego, położony najbliżej w przestrzeni cech (na rysunku - trójkąt)

Przykład. Metoda NN jest chętnie stosowana w automatyzacji diagnostyki medycznej, gdyż jest prosta pojęciowo i odpowiada intuicyjnie akceptowalnej zasadzie wskazywania - jako przypuszczalnej diagnozy - tej choroby, na którą cierpiał pacjent mający najbardziej podobne objawy do aktualnie badanego.

Algorytm tej metody opiszemy, stosując powszechnie przyjmowaną w literaturze informatycznej notację *pascalopodobną*. Notacja ta zakłada podporządkowanie zapisu regułom zbliżonym do obowiązujących w języku Pascal, jednak zapis nie jest w ścisłym sensie programem w Pascalu, ponieważ wiele mało istotnych a uciążliwych szczegółów algorytmu zastępuje się słownymi omówieniami. Przy prezentacji tego algorytmu przyjmuje się następujące założenia: zdefiniowane są następujące tablice i zmienne oraz funkcje (w nawiasach podano oznaczenia używane we wzorach):

numclass - liczba rozpoznawanych klas (L),

dim - wymiar przestrzeni cech (n),

num - liczba obiektów ciągu uczącego (N),

sampl[1 .. num][1 .. dim + 1] - ciąg uczący (U),

rec - identyfikator rozpoznanego obrazu (i),

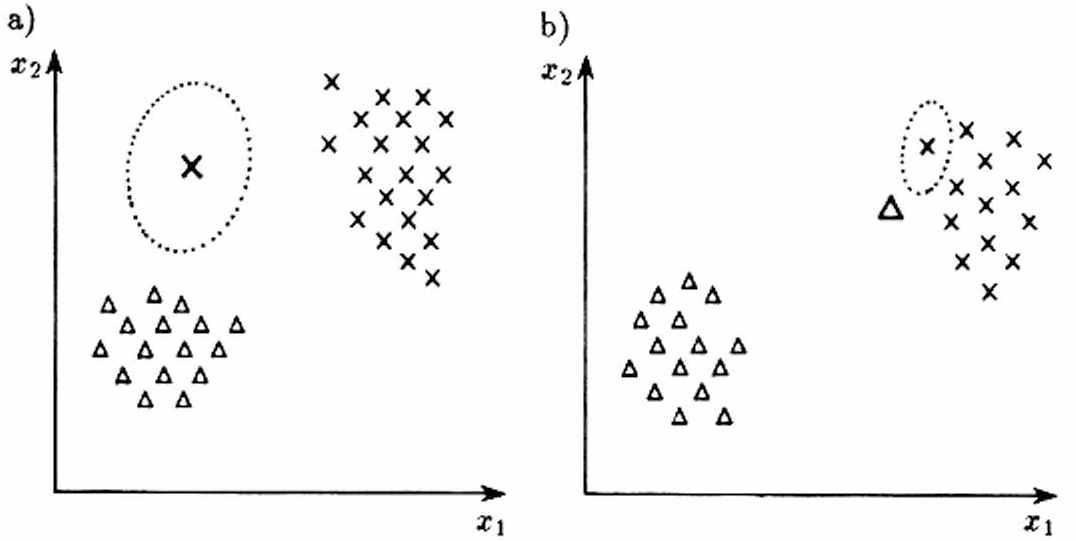
obj[1 .. dim] - rozpoznawany obiekt (x).

dist(sampl [k], obj) - funkcja podająca odległość (ρ) między k-tym elementem ciągu uczącego a rozpoznawanym obiektem

```
procedure NNrec (obj, var rec);
begin
      rec := 0; min := MaxReal;
      for k := 1 to num do
        if dist(sampl[k], obj) < min then
          begin
            min := dist(sampl[k], obj);
            rec := sampl[k] [dim + 1];
          end
end
```

4.3. Metoda αNN

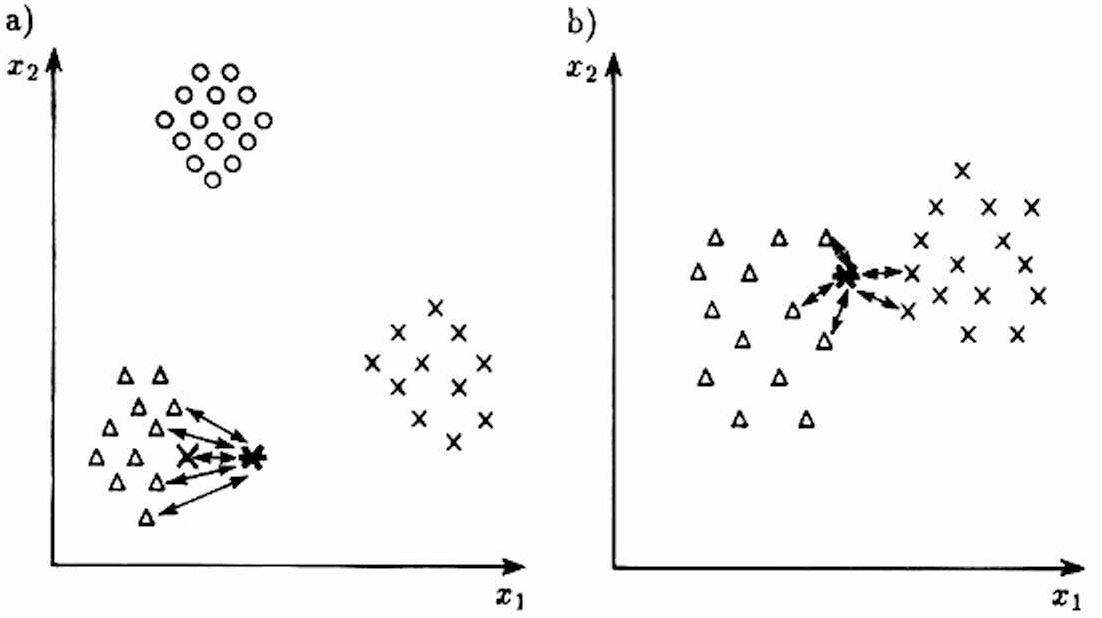
Czysta metoda NN (tak, jak ją przedstawiono) ma liczne wady. Jedną z nich jest jej duża wrażliwość na błędy ciągu uczącego U (rys. 4.3). Istotnie, jeśli błędnie określona zostanie przynależność i^k chociaż jednego elementu $\underline{x}^k$ ciągu uczącego U, to wówczas całe jego otoczenie będzie błędnie klasyfikowane.



Rys. 4.3. Jeśli położenie (a) lub sklasyfikowanie (b) chociaż jednego obiektu ciągu uczącego jest błędne, to wówczas w pewnym obszarze przestrzeni cech mogą być podejmowane błędne decyzje

Przykład. W zadaniu automatycznej wizualnej identyfikacji elementów podawanych przez robot do sterowanego cyfrowo centrum obróbczego przewidywano rozpoznawanie metodą NN z pokazem wzorców rozpoznawanych elementów poprzedzającym proces właściwego rozpoznawania (aby zapewnić elastyczność systemu i jego zdolność do pracy z obiektami o różnych kształtach). Aby uniezależnić rozpoznawanie od powiększenia obrazu (obiekty mogły być bliżej albo dalej od kamery) a także od kąta obrotu (obiekty nadjeżdżały na transporterze w dowolnych, przypadkowych położeniach), rozpoznawanie oparto na dwóch cechach: współczynniku kształtu konturu obiektu (stosunek powierzchni obiektu do kwadratu jego obwodu) i na liczbie występujących w nim otworów.

Rozpoznawaniu poddano łatwo odróżnialne obiekty: owalne pierścienie, prostokątne podkładki z dwoma otworami i trójkątne gładkie pokrywy. Po uruchomieniu systemu doszło do uszkodzenia obrabiarki, ponieważ duża trójkątna pokrywa została przez robot umieszczona w małej szlifierce do pierścieni. Analiza wykazała, że robot zapamiętał wzorzec pokrywy w tej samej klasie, co wzorzec pierścienia, ponieważ odblask reflektora oświetlającego na gładkiej powierzchni pokrywy układ rozpoznający potraktował jako „otwór", zaś liczba otworów silniej wpływała na wartość używanej metryki niż współczynnik kształtu. W rezultacie jeden błędnie zapamiętany wzorzec wygenerował całą klasę fikcyjnych „trójkątnych obiektów z jednym otworem", rozważanych jako pierścienie.



Rys. 4.4. Metoda αNN zapobiega błędom wynikającym z pomyłek w ciągu uczącym (a), ale ogranicza czułość metody (b)

Aby efekt ten ograniczyć, wprowadza się metodę αNN (rys. 4.4). W metodzie tej stosuje się parametr α, którego wybór determinuje własności metody. Parametr ten jest wybierany arbitralnie, jednak w ten sposób, aby

$$\alpha \ll \min_{i \in I} N^i. \tag{25}$$

W praktyce α jest małą liczbą całkowitą.

Przykład. W zadaniu automatycznego rozpoznawania samogłosek dokonywano prób rozpoznawania metodą αNN *przy różnych wartościach parametru* α. Uzyskane wyniki przedstawiono w tabeli 4.2. Jak widać, zarówno za mała, jak i za duża wartość α powoduje pogorszenie jakości rozpoznawania (rys. 4.5).

Po określeniu α i pojawieniu się obiektu rozpoznawania $\underline{x}$ obliczane są wartości odległości tego obiektu od wszystkich obiektów ciągu uczącego $\rho(\underline{x}, \underline{x}^k)$, $k = 1, 2, \ldots, N$. Następnie dokonuje się uporządkowania ciągu uczącego według rosnących odległości, wprowadzając numery $\nu = 1, 2, \ldots, n$:

$$U = \{\langle \underline{x}^\nu, i^\nu \rangle, \qquad \varphi = 1, 2, \ldots, N\}, \tag{26}$$

przy czym

$$\forall_{\nu,\mu} [\nu > \mu \Rightarrow \rho(\underline{x}, \underline{x}^\nu) > \rho(\underline{x}, \underline{x}^\mu)]. \tag{27}$$

Tabela 4.2. Zależność prawdopodobieństwa poprawnego rozpoznania od wartości parametru α w metodzie αNN

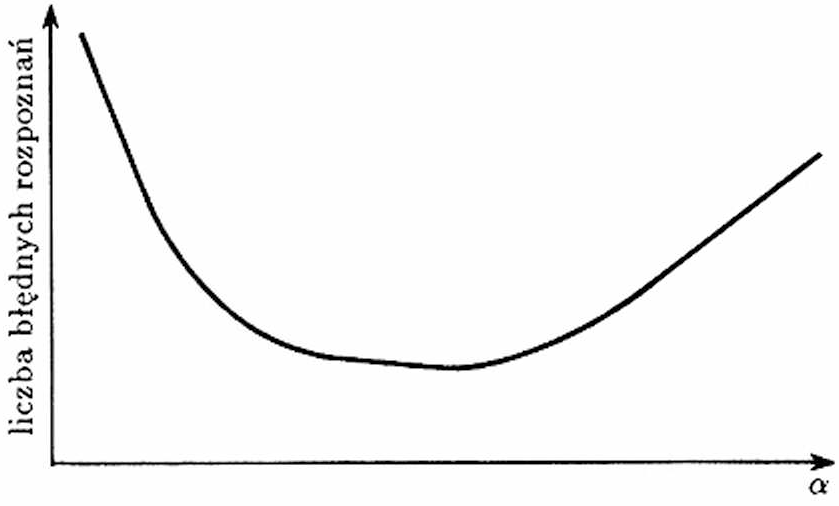
Wartość α	Prawdopodobieństwo poprawnego rozpoznania
1	0,923
2	0,962
3	0,974
4	0,963
5	0,969

Następnie wybiera się α początkowych obiektów ciągu, tworząc podzbiór

$$U^{\alpha} = \{\langle \underline{x}^{\nu}, i^{\nu}\rangle, \qquad \nu = 1, 2, \ldots, \alpha\} \tag{28}$$

rozbijany na podzbiory (ewentualnie puste) związane z poszczególnymi klasami

$$U^{\alpha,i} = \{\langle x^{\nu}, i^{\nu}\rangle, \qquad \nu < \alpha \wedge i^{\nu} = i\}. \tag{29}$$



Rys. 4.5. Przybliżona zależność liczby pomyłek przy rozpoznawaniu metodą αNN od parametru α wykazuje, że niekorzystne jest przyjmowanie zarówno zbyt dużego jak i zbyt małego α

Funkcje przynależności dla metody αNN można teraz wyznaczyć na podstawie liczebności podzbiorów $U^{\alpha,i}$

$$C^{i}(\underline{x}) = \#U^{\alpha,i}; \qquad i = 1, 2, \ldots, L. \tag{30}$$

Zasadę tej metody przedstawimy – jak poprzednio – w postaci pseudopascalowskiego algorytmu. Założenia dla tego algorytmu są takie same, jak przytoczone w podrozdziale 4.2 z następującymi uzupełnieniami:

alpha – zmienna określająca parametr α,

tab[1..num][1..2] – tablica odległości,

sort(tab) – funkcja dokonująca sortowania tablicy **tab**,

fun[1..numclass] – tabela wartości funkcji przynależności,

pointmax(fun) – funkcja wskazująca numer klasy, dla której wartość funkcji przynależności jest maksymalana.

```
procedure AlphaNNrec (obj, var rec);
begin
      fun := 0;          {wyzerowanie całej tablicy}
      for k := 1 to num do
        begin
          tab[k][1] := dist(sampl[k], obj);
          tab[k][2] := sampl[k][dim+1];
        end
      sort(tab);
      for k := 1 to alpha do
        fun[tab[k][2]] := fun[tab[k][2]] + 1;
      rec := pointmax(fun);
end
```

4.4. Metoda $j_N NN$

Pokrewne własności do metody αNN ma algorytm $j_N NN$, którego istota polega na określaniu przynależności nieznanego obiektu $\underline{X}$ do tej klasy i, do której należy j_N-ty w kolejności element zbioru U uporządkowanego według reguł określonych wzorami (26) i (27). Formalnie można zapisać, że funkcje przynależności są określone wzorem:

$$C^i(\underline{x}) = \delta_{ij_{N^i}}, \tag{31}$$

gdzie funkcja zgodności δ (*delta Kroneckera*) jest określana wzorem:

$$\delta = \begin{cases} 0 & \text{dla} \quad \mu \neq \nu, \\ 1 & \text{dla} \quad \mu = \nu, \end{cases} \tag{32}$$

zaś zapis i^{j_N} oznacza wskaźnik przynależności j_N-tego elementu uporządkowanego ciągu uczącego (26). W literaturze można znaleźć dowody identyczności własności algorytmu $j_N NN$ i procedury αNN dla

$$j_N = \text{ent}\left(\frac{\alpha+1}{2}\right), \tag{33}$$

gdzie ent(μ) oznacza część całkowitą liczby rzeczywistej μ. W praktyce obserwuje się wyraźnie gorsze działanie algorytmu $j_N NN$ od działania algorytmu αNN i dlatego jego prezentacja w niniejszej książce ma charakter jedynie porządkowy([2]).

Przykład. W pracy doktorskiej L.Kota [59] badano - między innymi - jakość rozpoznawania spółgłosek szumowych([3]). Wykazano, że poprawność rozpoznawania głoski **Z** uzyskiwana za pomocą algorytmu NN wyraża się odsetkiem 71,4% poprawnych identyfikacji. Algorytm αNN dla $\alpha = 5$ pozwolił uzyskać aż 91,0% poprawnych odpowiedzi, natomiast wartość uzyskana za pomocą algorytmu $j_N NN$ wynosiła 61,9% poprawnych rozpoznań.

Na zakończenie jak zwykle podamy algorytm metody $j_N NN$. Wymaga on tylko wprowadzenia jednej dodatkowej zmiennej w stosunku do algorytmu opisanego w poprzednim podrozdziale:

point - wybrana wartość j_N

```
procedure jN_NNrec (obj, var rec);
begin
      fun := 0;          {wyzerowanie całej tablicy}
      for k := 1 to num do
        begin
          tab[k][1] := dist(sampl[k], obj);
          tab[k][2] := sampl[k][dim+1];
        end
      sort(tab);
      rec := tab[point] [2];
end
```

([2]) Łatwo zauważyć, że metody αNN i $j_N NN$ sprowadzają się do podstawowej reguły NN dla $\alpha = 1$ ($j_N = 1$).

([3]) Przy przygotowywaniu przykładu wybrano tę klasę głosek, ponieważ dawała podstawy do porównań.

4.5. Podsumowanie

Algorytmy minimalnoodległościowe są chętnie stosowane w praktyce ze względu na prostotę i intuicyjność, a także dlatego, że dają stosunkowo dobre wyniki (mały wskaźnik błędnych rozpoznań). Są one jednak kosztowne w realizacji, gdyż wymagają archiwizowania całego ciągu uczącego U w pamięci systemu rozpoznającego oraz czasochłonnego obliczenia odległości ρ rozpoznawanego obiektu $\underline{x}$ od wszystkich elementów ciągu uczącego $\underline{x}^k$, co w ogólnym przypadku zmusza do stosowania dużych mocy obliczeniowych lub godzenia się z długim czasem rozpoznawania.

Przykład. Długość ciągu uczącego N ma wpływ nie tylko na zajętość pamięci i na czas obliczeń, ale również wpływa na dokładność rozpoznawania. Konkretne wyniki, uzyskane przy rozpoznawaniu samogłosek z wykorzystaniem metody NN przedstawiono w tabeli 4.3.

Tabela 4.3. Wpływ długości ciągu uczącego na dokładność rozpoznawania

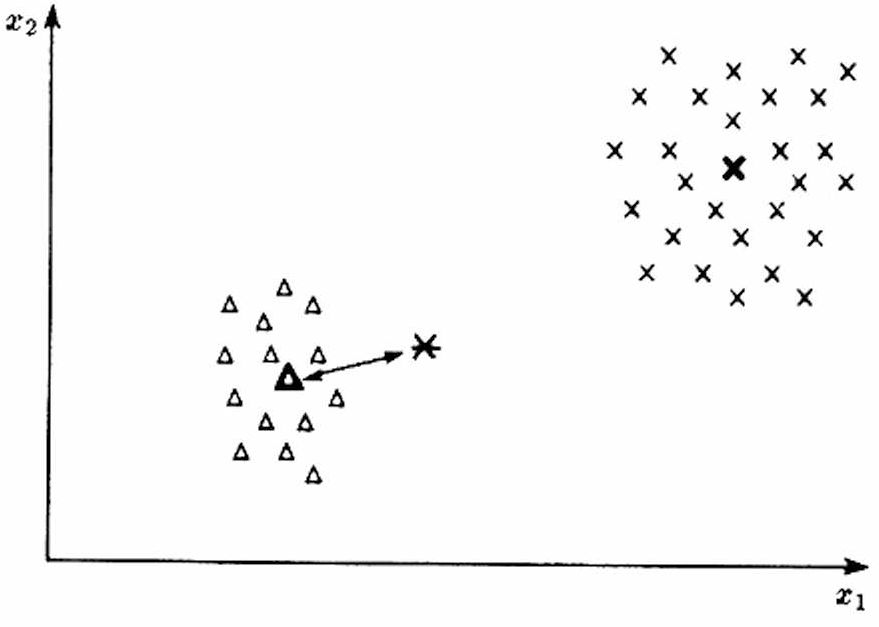
Długość ciągu uczącego	Procent poprawnie rozpoznanych głosek
13	0,923
30	0,933
46	0,935
61	0,918
101	0,980
203	0,961

5. METODY WZORCÓW

5.1. Metoda uogólnionych wzorców i otoczeń kulistych

W metodach omawianej grupy odwzorowanie C sprowadza się do określenia *zawierania* się nieznanego obiektu x w obszarze wzorca i-tej klasy W^i ($W^i \subset X$) (rys. 5.1). Funkcję przynależności można tu formalnie zapisać w postaci:

$$C^i(\underline{x}) = \begin{cases} 1 & \text{dla} \quad \underline{x} \in W^i, \\ 0 & \text{w przeciwnym przypadku.} \end{cases} \tag{34}$$

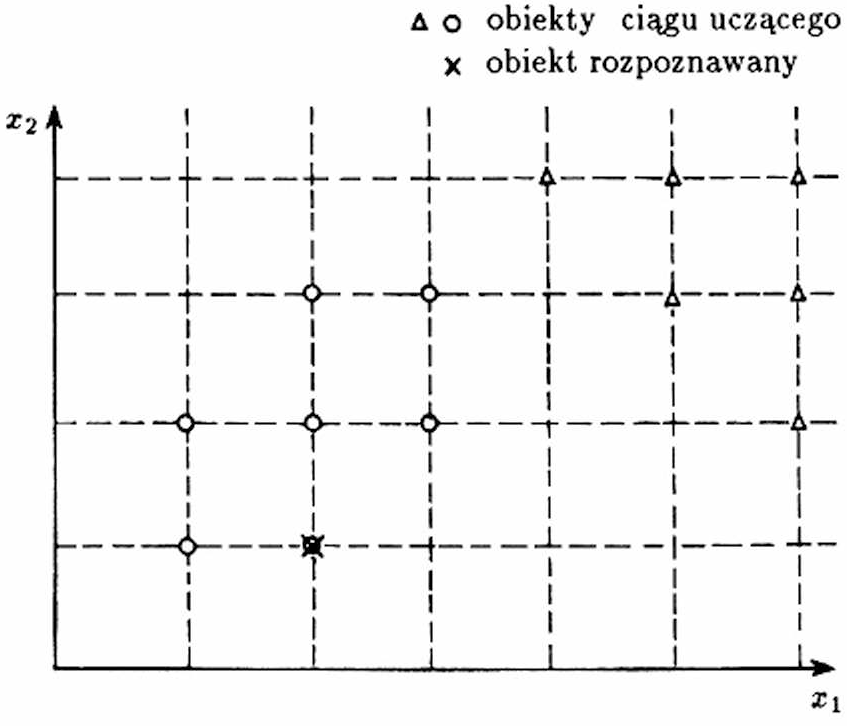


Rys. 5.1. Ilustracja pojęcia wzorca

Naturalnie, kluczowym problemem staje się przy tym sposób definicji wzorca W^i. W najprostszym przypadku *wzorzec* można utożsamiać z odpowiednim *podzbiorem* ciągu uczącego (porównaj (19) i (20))

$$W^i = U^i. \tag{35}$$

Reguła decyzyjna oparta na funkcji przynależności (34) sprowadza się wówczas do reguły *pokrycia punktów*, której zaletą może być w ogólnym przypadku mały procent błędnych rozpoznań. Przy odpowiedniej strukturze przestrzeni cech X (*gruba* dyskretyzacja wartości cech x_j) liczba decyzji neutralnych i_o (braków odpowiedzi – porównaj (13), (14) i (15)) może być zadowalająco mała (rys. 5.2).



Rys. 5.2. Przy dyskretyzowanych cechach prawdopodobieństwo ropoznania metodą pokrycia punktów jest bardzo duże

Przykład. W zadaniu diagnostyki medycznej często występują cechy przyjmujące tylko kilka wartości: wymioty występują lub nie, gorączka może być silna, słaba lub może wcale nie występować, liczbę chorobotwórczych drobnoustrojów w badaniach mikrobiologicznych oznacza się w skali:

$-$	(brak),	$+$	(obecne),
$++$	(liczne),	$+++$	(bardzo liczne).

Przy takiej dyskretyzacji przestrzeni cech jest nader prawdopodobne pokrycie się punktów ciągu uczącego z cechami diagnozowanego pacjenta.

Zaletą tej metody jest wyjątkowo prosty algorytm, który w notacji pascalopodobnej może być zapisany następująco:

```
procedure ident(obj, var rec);
begin
    k := 0;
    repeat
      k := k + 1
    until k > num or dist(sampl[k], obj) = 0;
    if k > num then rec := 0 else rec := sampl[k][dim + 1];
end
```

Obiekty użyte w tym algorytmie zdefiniowano w poprzednim rozdziale. W ogólnym przypadku omówiona metoda jest niepraktyczna, ponieważ występuje ogromny odsetek braków decyzji i_o, co powoduje konieczność modyfikacji pojęcia wzorca W^i w stosunku do definicji (35). Możliwe jest to między innymi poprzez wprowadzenie techniki *otoczeń kulistych* (rys. 5.3). (Pojęcie kuli musi być tu rozumiane w sposób uzależniony od przyjętej definicji metryki ρ przestrzeni X([1]).)

$$W^i = \{\underline{x} : \exists\, \underline{x}^{i,k} \in U^i [\rho(\underline{x}, \underline{x}^{i,k}) < \varepsilon^{i,k}]\}. \tag{36}$$

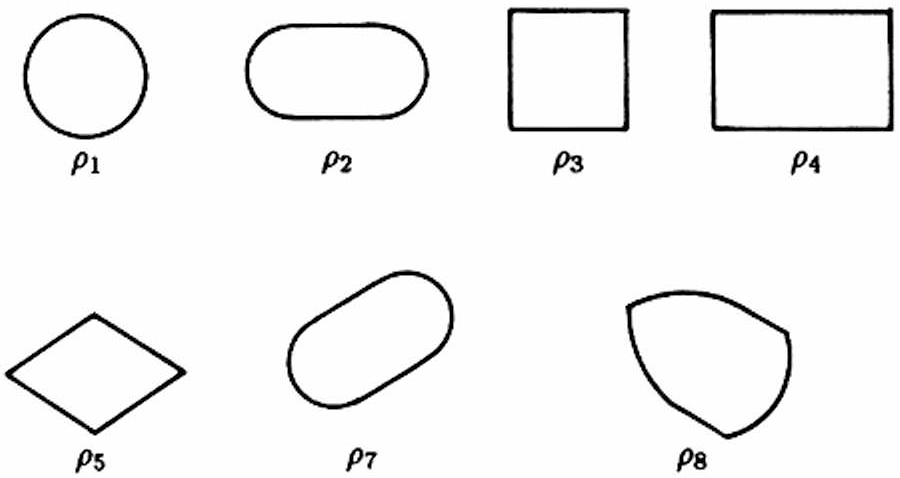
W definicji (36) istotną rolę odgrywają promienie otoczeń kulistych $\varepsilon^{i,k}$. Ich wybór determinuje własności metody, przy czym oczywiście każdy wzrost wartości $\varepsilon^{i,k}$ prowadzi do ograniczenia liczby decyzji odmownych i_o, ale równocześnie zwiększa odsetek decyzji błędnych. W najprostszym przypadku przyjmuje się stałą wartości $\varepsilon^{i,k}$:

$$\forall_{i \in I} [\forall_{k \in [1, N^i]} [\varepsilon^{i,k} = \varepsilon = \text{const}]]. \tag{37a}$$

W przeciwnym przypadku można ustalić każdą wartość $\varepsilon^{i,k}$ na innym poziomie, na przykład na podstawie reguły:

$$\varepsilon^{i,k} = \frac{1}{2} \left[\min_{\underline{x}^\mu \notin U^i} \rho(\underline{x}^{i,k}, \underline{x}^\mu) \right]. \tag{37b}$$

([1]) Patrz także Dodatek 1.



Rys. 5.3. Otoczenie kuliste ma kształt zależny od przyjętej metryki (oznaczenia ρ_1, ρ_2 odpowiadają oznaczeniom w Dodatku 1)

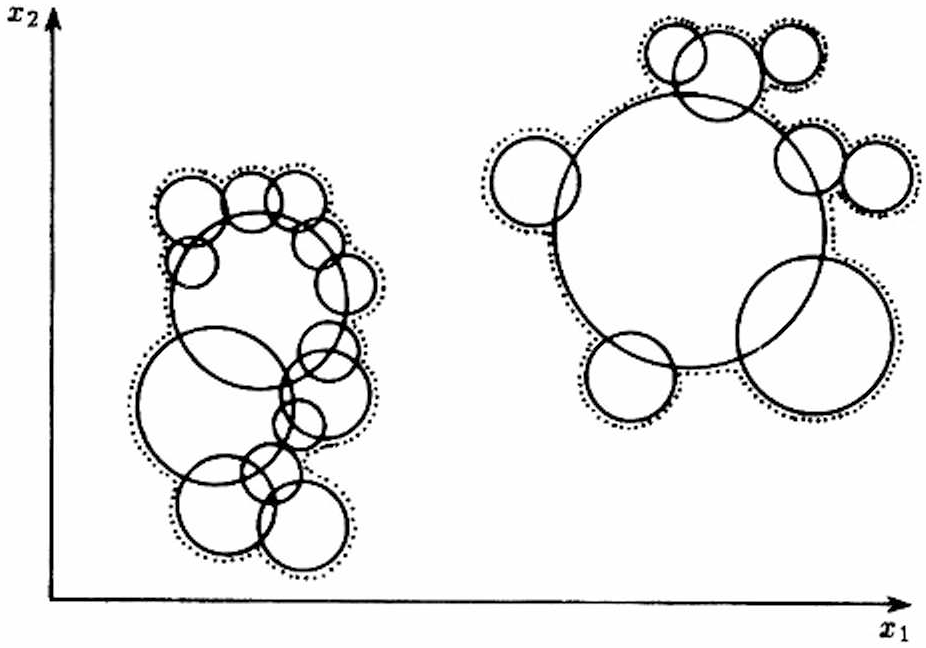
Wartość ε we wzorze (37a) może być wybierana według reguły

$$\varepsilon = \min_{i \in I}\left(\min_{k \in [1, N^i]} \varepsilon^{i,k}\right). \tag{38}$$

Przyjęcie zasady (37b) powoduje polepszenie jakości (niezawodności i wiarygodności) rozpoznawania (rys. 5.4), ale prowadzi do dodatkowego obciążenia pamięci systemu rozpoznającego, gdyż konieczne jest pamiętanie wszystkich wartości $\varepsilon^{i,k}$. Przyjęcie reguły (37a) oszczędza pamięć i ogranicza czas obliczeń, ale zwiększa procent odpowiedzi odmownych i_o.

Przykład. Przy identyfikacji obiektów obserwowanych za pomocą radaru stosuje się - obok innych metod - technikę rozpoznawania obrazów. Na podstawie zapamiętanych zestawów cech echa radarowego wszystkich znanych obiektów (określone typy obiektów latających, chmury, stada ptaków itp.) procesor komputera wspomagającego pracę kontrolera przyjmuje wstępne hipotezy na temat tożsamości określonych obiektów rejestrowanych przez radar w nadzorowanym obszarze. Stosuje się przy tym zawsze pewne tolerancje: wartości parametrów echa identyfikowanego obiektu uznane zostają za zgodne z zapamiętanymi wartościami wzorca również wtedy, gdy nie są dokładnie identyczne, lecz różnią się o pewną wartość - oszacowaną z góry na podstawie możliwych błędów pomiarowych.

Osobnym problemem przy stosowaniu metod otoczeń kulistych jest nadmiarowość ciągu uczącego. Istotnie, łatwo sobie wyobrazić, że przyj-



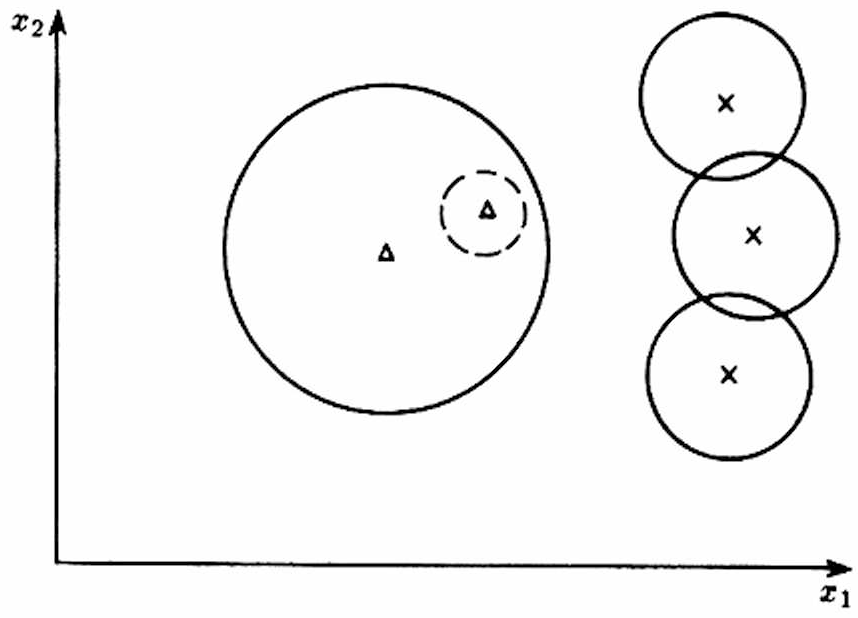
Rys. 5.4. Otoczenia kuliste o różnych promieniach pozwalają bardzo dokładnie odwzorować kształty obszarów o różnej topografii

mując regułę (37b) możemy doprowadzić do sytuacji (rys. 5.5) opisanej wzorem:

$$\exists \underline{x}^{i,\mu} \in U^i \left[\exists \underline{x}^{i,k} \in U^i \quad [\rho(\underline{x}^{i,k}, \underline{x}^{i,\mu}) + \varepsilon^{i,\mu} < \varepsilon^{i,k}]\right]. \tag{39}$$

W takim przypadku punkt ciągu uczącego $\underline{x}^{i,\mu}$ znika wraz ze swoim otoczeniem kulistym w otoczeniu punktu $\underline{x}^{i,k}$ i nie wnosi żadnej informacji w sensie funkcji przynależności (34). Ze względu na optymalizację zajętości pamięci i czasu obliczeń, punkty $\underline{x}^{i,\mu}$ spełniające zależność (19) powinny być usuwane z ciągu U. Istnieją ogólne algorytmy takiego usuwania (por. [1] oraz [2]), są one jednak zbyt złożone, by je tu przytaczać.

Przykład. W systemach gromadzenia danych naukowych pochodzących z badań różnych laboratoriów analizujących ten sam globalny temat pojawia się z całą ostrością problem usuwania informacji dublujących się, które mogą być identyczne pod względem treści, lecz mogą znacznie różnić się swoją formą. Przykładem tego typu systemów mogą być międzynarodowe banki danych, gromadzące informacje na temat kodu genetycznego człowieka, lub system LUNAR,



Rys. 5.5. Otoczenia różnych punktów mogą się niekiedy pochłaniać

w którym zestawiano wyniki międzynarodowych badań próbek gruntu księżycowego. Stosowane i rozwijane metody eliminacji powtórzeń i stwierdzeń bliskoznacznych, wykorzystywane w tych systemach, bazują często na osiągnięciach teorii rozpoznawania obrazów.

Przytoczmy algorytm metody otoczeń kulistych. Niech:

radius[1 .. num] – promienie (ε) otoczeń związnych z kolejnymi obiektami ciągu uczącego

```
procedure ballrec (obj, var rec);
begin
      k := 0;
      repeat
        k := k + 1
      until k > num or dist(sampl[k], obj) <= radius[k];
      if k > num than rec := 0 else rec := sampl[k][dim + 1];
end
```

5.2. Metoda NM

Usuwając kolejne punkty z ciągu uczącego, doprowadzamy niekiedy do sytuacji, w której każda klasa $i = 1, 2, \ldots, L$ reprezentowana jest przez jeden obiekt $\underline{M}^i = (m^i, m^i, \ldots, m^i)$, zwany zwykle obiektem modalnym lub – krótko – *modą* (rys. 5.6). Dysponując właściwie dobranymi modami dla każdej klasy, mamy możność wyraźnego uproszczenia i przyspieszenia procesu rozpoznawania poprzez stosowanie metody *najbliższej mody*. (Przez analogię do metody NN metoda najbliższej mody bywa denotowana symbolem NM). Funkcja przynależności w tej metodzie może być określona w prosty sposób jako

$$C^i(\underline{x}) = \frac{1}{\rho(\underline{x}, M^i) + \varepsilon}, \qquad i = 1, 2, \ldots, L, \tag{40}$$

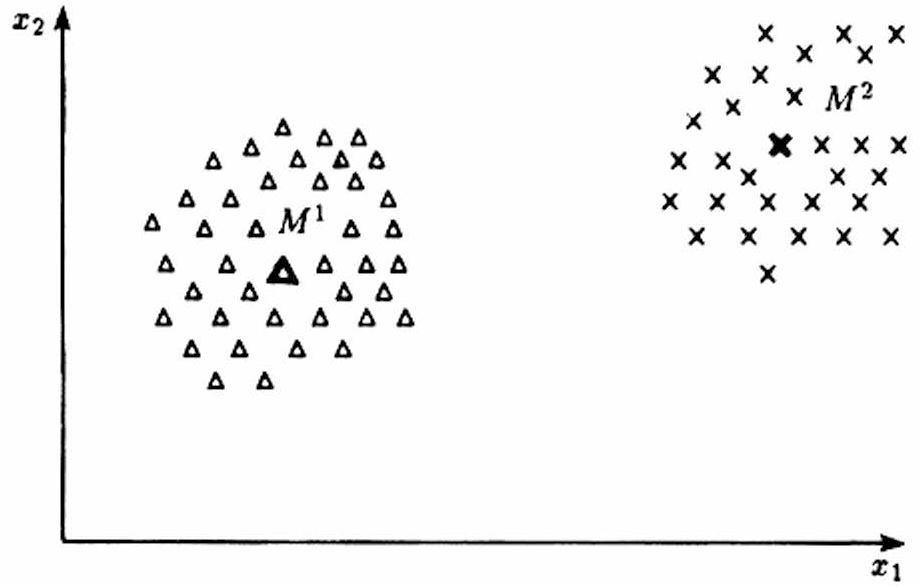
co gwarantuje zarówno niewielkie zajęcie pamięci komputera, jak i krótki czas obliczeń. Podstawowa trudność przy stosowaniu metody NM polega na złożonej i nie zawsze skutecznej technice wyszukiwania mody M^i (dla $i = 1, \ldots, L$) na drodze usuwania *nadmiarowych* obiektów ciągu uczącego. Niekiedy procedurę wyszukiwania mody upraszcza się, stosując operację uśredniania:

$$m_j^i = \frac{1}{N_i} \sum_{k=1}^{N_i} x_j^{i,k}, \qquad i = 1, \ldots, L; \qquad j = 1, \ldots, n. \tag{41}$$

Stosując operator wartości oczekiwanej $\mathsf{E}\{\cdot\}$, możemy wzór (41) zapisać prościej jako:

$$m_j^i = \mathsf{E}_k\{x^{i,k}\},$$

gdzie wskaźnik k przy symbolu E sygnalizuje względem jakiej numeracji prowadzono uśrednianie. Powstałe w ten sposób mody często bywają wystarczająco dokładnie wyznaczone (rys. 5.7), aby rozpoznawanie według reguły (40) z ich pomocą dawało zadowalające rezultaty, jakkolwiek m.in. w pracy [4] podano wiele przykładów błędnego funkcjonowania algorytmów opartych na wzorach (40) i (41) w przypadku „złośliwego" doboru ciągu uczącego U lub niekorzystnej topografii obszarów wyznaczanych w przestrzeni X przez obiekty poszczególnych klas (rys. 5.8). Mimo jednoznacznej wymowy tych przykładów można znaleźć argumenty wskazujące na



Rys. 5.6. Zastępowanie ciągów uczących klas obiektami modalnymi M jest jedną ze skutecznych metod redukcji pracochłonności rozpoznawania

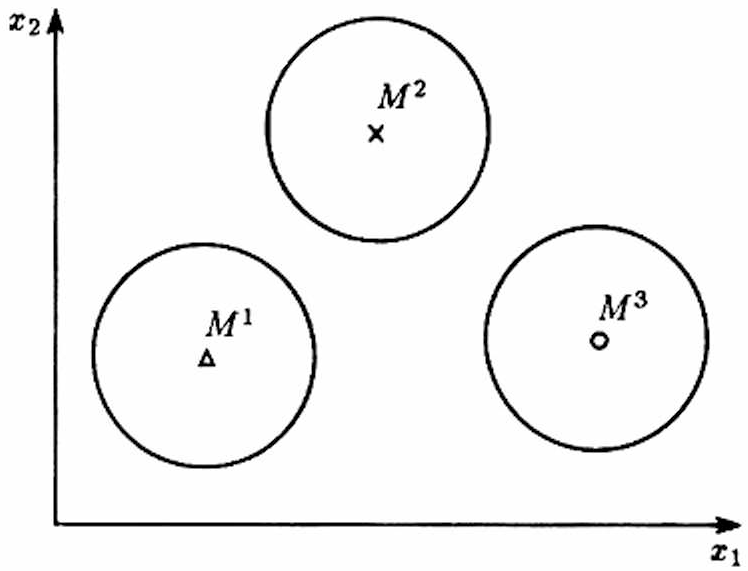
celowość stosowania metody (40), (41) w obszernej klasie zadań rozpoznawania. Niekiedy istnieje bowiem abstrakcyjny idealny wzorzec $\underline{\widetilde{M}}^i$ każdej klasy, do którego dążą, jakkolwiek w sposób niedoskonały, wszystkie konkretne realizacje obiektów $\underline{x}^{i,k}$ z danej klasy.

Przykład. Sytuacja tego typu ma miejsce w przypadku rozpoznawania rękopisów. Każda litera pisana dowolnym charakterem pisma jest zniekształconym naśladownictwem wzorca z podręcznika kaligrafii lub elementarza.

Jeżeli założymy, że na płaszczyźnie cech stanowiących bazę przestrzeni X zniekształcenie polega na dodaniu do wektora wzorca (zbioru cech obrazu dla klasy) $\underline{\widetilde{M}}^i$ wektora przypadkowego $\underline{\mu}^{i,k}$ o zerowej wartości oczekiwanej i ograniczonej wariancji wszystkich składowych:

$$\underline{x}^{i,k} = \underline{\widetilde{M}}^i + \underline{\mu}^{i,k}, \tag{42}$$

$$\forall_{i \in I}\left[\mathsf{E}_k\{\underline{\mu}^{i,k}\} = \underline{0} \wedge \sum_{\nu=1}^{n} \mathsf{E}_k\{(\mu_\nu^{i,k})^2\} < \infty\right], \tag{43}$$



Rys. 5.7. Przyjecie mody M jako środka ciężkości (średniej) obiektów rozważanych klas bywa bardzo dobrym rozwiązaniem w przypadku klas o regularnych i stosunkowo prostych kształtach

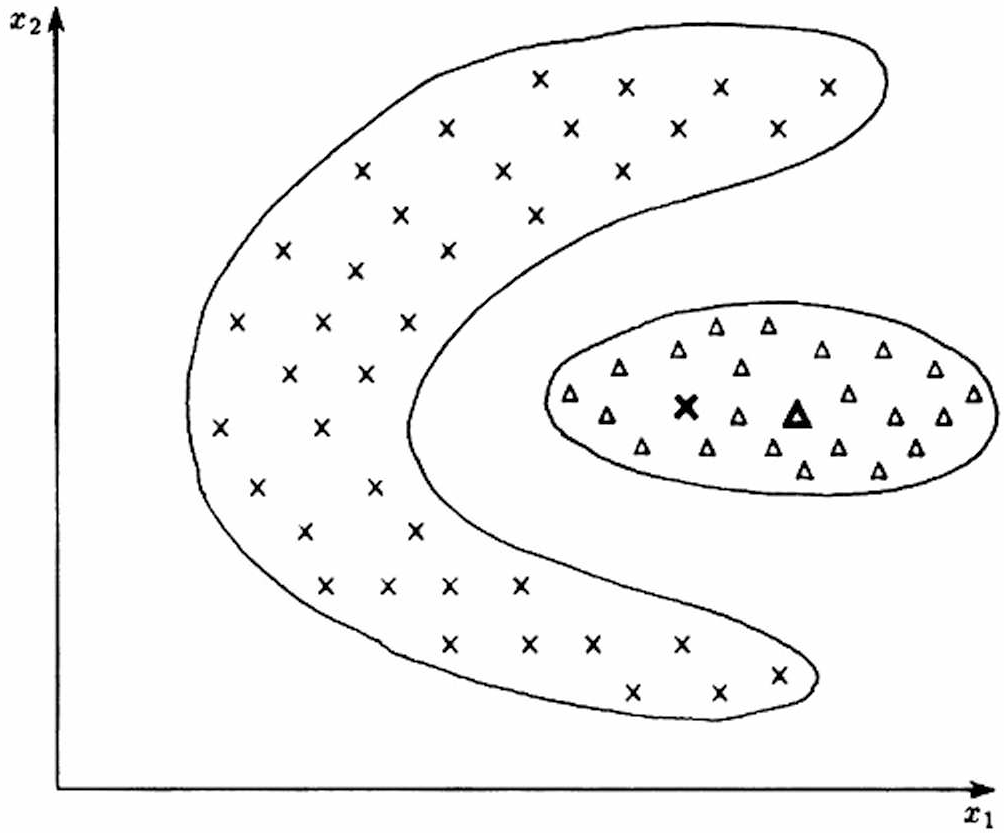
to wówczas wzór (41) określa sposób znalezienia nieobciążonego najefektywniejszego estymatora([2]) $\underline{M}^i$ idealnego wzorca $\underline{M}^i$.

Przykład. Metody rozpoznawania obrazów stosowane są niekiedy do oceny ekonomicznych wyników działania przedsiębiorstw. Wzorce M^i mogą być wówczas wyznaczane metodami ekonometrycznymi.([3])

Przytoczone rozumowanie zawiera dużo trudnych do sprawdzenia założeń (na przykład $\mathsf{E}_k\underline{\mu}^{i,k} = 0$) i w ogólnym przypadku jest nieprzydatne. Mimo to prostota i zalety użytkowe metody rozpoznawania danej wzorami (40) i (41) zachęcają do prób jej praktycznego stosowania. Powinien to ułatwić następujący algorytm, w którym (poza obiektami znanymi z poprzedniego rozdziału) należy wprowadzić macierz wartości modalnych jako tablicę

([2]) Estymatorem określonego parametru statystycznego nazywamy wartość obliczoną na podstawie pewnej liczby obserwacji realizacji zmiennych losowych. Wartość ta jest także zmienną losową (jako funkcja zmiennych losowych), przy czym, jeśli wartość oczekiwana jej błędu wynosi 0 , to estymator nazwiemy nieobciążonym, a jeśli jej wariancja jest minimalna, to dodajemy przymiotnik „najefektywniejszy".

([3]) Obszerniejszą dyskusję tych zagdanień zawarto w skrypcie [60].



Rys. 5.8. Przykłady klas, dla których średnia nie jest dobrym wzorcem dla całej klasy

mode [1 .. numclass][1 .. dim] – wartości modalne ($\underline{M}^i$)

i zmodyfikować funkcję **dist**, której pierwszy parametr ma obecnie (jako tablica) wymiar **dim** a nie (dim + 1) jak poprzednio. Tę zmodyfikowaną funkcję nazwiemy **dist1**

```
procedure NMrec (obj, var rec);
begin
      rec := 0; min := MaxReal;
      for i := 1 to numclass do
      if dist1(mode[i], obj) < min then
        begin
          rec := i;
          min := dist1(mode[i], obj)
        end
end
```

6. METODY APROKSYMACYJNE

6.1. Postawienie zadania

Omawiane poprzednio metody charakteryzowało dążenie do zaproponowania algorytmu rozpoznawania opierającego się na rozmaitych intuicjach geometrycznych, związanych z położeniem w *przestrzeni cech* punktu odpowiadającego rozpoznawanemu obiektowi $\underline{x}$ w stosunku do zbioru punktów $\underline{x}^{i,k}$ tworzących ciąg uczący. Możliwe jest również całkowicie inne podejście. Jak wiadomo, poszukujemy odwzorowania C (por. (19)) spełniającego określone założenia (por. (12)) i oczywiście zakładamy, że odwzorowanie takie *istnieje*. Możliwe jest więc następujące rozumowanie.

Funkcję przynależności $C^i(\underline{x})$ można (przy łatwych do spełnienia założeniach) rozwinąć w szereg względem ustalonej rodziny funkcji φ

$$\forall_{i\in I}\left[\forall_{\underline{x}\in X}\left[C^i(\underline{x}) = \sum_{\nu=0}^{\infty} V_\nu^i \varphi_\nu(\underline{x})\right]\right], \tag{44}$$

gdzie użyte funkcje bazowe tworzą uporządkowaną rodzinę

$$\Phi = \langle \varphi_1(\underline{x}), \varphi_2(\underline{x}), \ldots \rangle. \tag{45}$$

Współczynniki rozwinięcia V_ν^i ($i = 1, \ldots, L$; $\nu = 1, 2, \ldots$), nazywane dalej wagami, wyznaczają konkretną funkcję $C^i(\underline{x})$. Będą one nadal głównym obiektem naszego zainteresowania, gdyż ustaliwszy rodzinę funkcji Φ, będziemy mogli zadanie wypracowania reguł decyzyjnych dla poszególnych klas zastąpić zadaniem wyznaczenia wartości wag V_ν^i ($i = 1, 2, \ldots, L$).

Aby postępowanie takie było efektywne, musimy ograniczyć zakres sumowania we wzorze (44). Załóżmy (co będzie przedmiotem dalszych rozważań), że funkcje przynależności $C^i(\underline{x})$ dobrze rozwijają się w szereg względem wybranej rodziny funkcji Φ, w wyniku czego wartości wag V^i dla $\nu > m$ mogą być pominięte

$$\forall_{i \in I} \left[\forall_{\nu > m} \left[V_\nu^i \leq \varepsilon \right] \right]. \tag{46}$$

W efekcie można stosować zapis

$$C^i(\underline{x}) = \sum_{\nu=0}^{m} V_\nu^i \varphi_\nu(\underline{x}), \tag{47}$$

implikujący bardzo wygodną z praktycznego punktu widzenia sytuację: do określenia wszystkich funkcji przynależności wystarczy wyznaczyć tylko Lm współczynników V_ν^i, które są też jedyną konieczną do zapamiętania informacją przy dalszym rozpoznawaniu. W stosunku do konieczności pamiętania całego ciągu uczącego U dla metod minimalnoodległościowych jest to duża oszczędność.

Przykład. W zadaniach analizy i rozpoznawania mowy rozważanych w pracy doktorskiej A. Izworskiego [61] stosowano ciąg uczący, obejmujący ponad 36 tysięcy próbek sygnału, będących wektorami w 96-wymiarowej przestrzeni cech. Do zapamiętania tego zbioru danych w przypadku używania metod minimalnoodległościowych konieczne były 4 megabajty pamięci. Tymczasem przy rozpoznawaniu wszystkich 40 fonemów języka polskiego z użyciem metod aproksymacyjnych konieczne było zapamiętanie zaledwie około 5 tysięcy współczynników V_ν^i, co wymagało niespełna 20 kB.

6.2. Problem wyboru funkcji bazowych

Zasadnicza trudność przy stosowaniu w praktyce wzoru (47) wiąże się z koniecznością wyboru rodziny funkcji Φ. Trudność ta wynika z potrzeby spełnienia kilku warunków. Po pierwsze, trzeba zapewnić warunki dla zastąpienia wzoru (44) wzorem (47), czyli – dobrą rozwijalność funkcji $C^i(\underline{x})$ (o których nie wiemy nic poza tym, że istnieją!) w szereg względem funkcji

rodziny Φ. Po drugie, wygodnie jest dysponować rodziną funkcji ortogonalnych lub jeszcze lepiej – ortonormalnych, to znaczy spełniających warunki

$$\forall_{\mu,\nu\in[1,m]}\left[\mu \neq \nu \Rightarrow \int\int \cdots \int_{n} \varphi_\mu(\underline{x}) \cdot \varphi_\nu(\underline{x})\, \mathrm{d}\underline{x} = 0\right], \tag{48}$$

$$\forall_{\mu,\nu\in[1,m]} \int\int \cdots \int_{n} \left[\varphi(\underline{x})\right]^2 \mathrm{d}\underline{x} = 1. \tag{49}$$

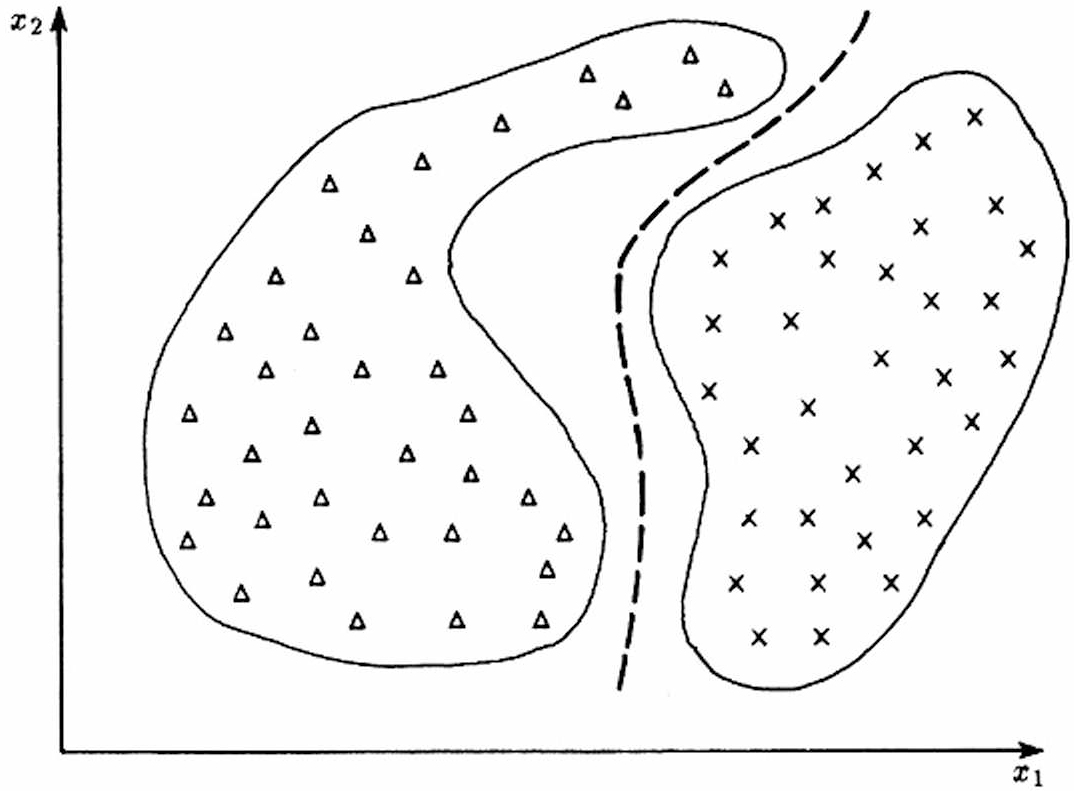
Ortonormalność funkji bazowych powoduje, że poszczególne wagi V_ν^i mogą być wyznaczane niezależnie, a błąd wyznaczenia jednej z nich nie wpływa na błędy wyznaczenia pozostałych. Powstaje jednak problem, jak znaleźć rodzinę ortonormalnych funkcji n zmiennych (argument $\underline{x}$ jest wektorem), gdyż dobrze znane są jedynie odpowiednie rodziny funkcji jednej zmiennej (na przykład rodziny funkcji trygonometrycznych, wykładniczych, wielomianów Czebyszewa itp.). Zanim podamy propozycje jednej z metod tworzenia rodziny funkcji Φ, musimy zastanowić się nad możliwością spełnienia warunku (46) i nad zapewnianiem zbieżności szeregu (47).

Niewiele wiedząc o charakterze funkcji przynależności $C^i(\underline{x})$, możemy jednak przypuszczać, że są to funkcje w miarę gładkie o małej *dziwaczności* (por. [5]). Pojęcie *dziwaczności*, ma charakter intuicyjny i nieformalny, ale można je sprecyzować na przykład w ten sposób, że funkcję będziemy uważać za tym bardziej dziwaczną, im więcej ma ekstremów w pewnym ustalonym przedziale zmienności swoich argumentów. Przypuszczenie o małej dziwaczności funkcji przynależności $C^i(\underline{x})$ wynika z intuicyjnego przekonania o regularnym kształcie klas w przestrzeni X, gdyż dobre odseparowanie obszarów należących do różnych klas pozwala na ich rozgraniczenie stosunkowo gładką hiperpowierzchnią w n-wymiarowej przestrzeni (por. rys. 6.1). Równanie hiperpowierzchni rozgraniczającej klasę μ od ν ma oczywiście postać:

$$C^\mu(\underline{x}) - C^\nu(\underline{x}) = 0. \tag{50}$$

Gładkość granic obszarów w przestrzeni X determinuje więc małą dziwaczność funkcji $C^i(\underline{x})$ i na odwrót.

Jeśli teraz rodzinę funkcji Φ wybierzemy w ten sposób, aby ze wzrostem numeru porządkowego ν harmonik $\varphi_\nu(\underline{x})$ rosła ich dziwaczność (własność taką mają liczne rodziny funkcji jednej zmiennej, na przykład wielomiany Czebyszewa), to wówczas warunek (47) będzie prostą konsekwencją oczywistego faktu, że do budowy funkcji w ograniczonym stopniu dziwacznej nie trzeba używać wysoce dziwacznych harmonik.



Rys. 6.1. Mało skomplikowanym kształtom obszarów obrazów w przestrzeni cech odpowiadają gładkie powierzchnie rozgraniczające te obszary

Postulaty pod adresem rodziny funkcji Φ można więc zebrać w następującej postaci: ma to być rodzina ortonormalnych funkcji n zmiennych o wzrastającej dziwaczności. Rodzinę taką można utworzyć na podstawie łatwo dostępnej rodziny funkcji jednej zmiennej o tych samych własnościach. Niech $f_\mu(z)$ $(\mu = 0, 1, 2, \ldots)$ będzie rodziną ortonormalnych funkcji jednej zmiennej o rosnącej dziwaczności. Wówczas można budować rodzinę Φ, korzystając z formuły kanonicznej

$$\varphi_\nu(\underline{x}) = \prod_{\eta=1}^{n} f_{\mu_\eta}(x_\eta), \tag{51}$$

w której spełniony musi być warunek

$$\sum_{\eta=1}^{n} \mu_\eta = \nu. \tag{52}$$

Naturalnie, podana formuła jest niejednoznaczna, gdyż pozwala wygenerować n harmonik rzędu 1 ($\nu = 1$), $n(n-1)/2$ harmonik rzędu 2 i ogólnie $n!/(\nu!(n-\nu)!)$ harmonik rzędu ν. To bogactwo może jednak być łatwo opanowane: można włączyć do rodziny Φ tylko niektóre harmoniki rzędu ν albo wszystkie, porządkując je według dowolnych arbitralnych kryteriów – jest to obojętne, gdyż w każdym przypadku otrzymamy potrzebną rodzinę ortogonalnych funkcji o rosnącej (lub przynajmniej nie malejącej) dziwaczności.

Przykład. Jako rodzinę funkcji jednej zmiennej o potrzebnych nam własnościach rekomendować można rodzinę funkcji trygonometrycznych

$$f_\mu(z) = \cos \mu \pi z \tag{53}$$

lub wielomianów Czebyszewa

$$f_\mu(z) = \frac{(z + \sqrt{z^2 - 1})^\mu + (z - \sqrt{z^2 - 1})^\mu}{2^\mu}, \tag{54}$$

albo Legendre'a

$$f_\mu(z) = \frac{1}{2^\mu \mu!} \frac{\mathrm{d}^\mu[(z^2 - 1)^\mu]}{\mathrm{d}z^\mu}. \tag{55}$$

Możliwe są także rozmaite inne funkcje – dobierane przez użytkownika według jego orientacji (lub domniemań) na temat kształtu linii granicznych obszarów w przestrzeni cech.

Na zakończenie rozważmy algorytm budujący funkcje bazowe $\varphi_\nu(\underline{x})$ na podstawie rodziny funkcji skalarnych $f_\mu(z)$ przy założeniu, że numery harmonik są zadane tabelą, co pozwala rozstrzygnąć problem niejednoznaczności przy numeracji funkcji $f_\mu(z)$. Przyjmiemy następujące nazwy nowych zmiennych i funkcji:

fi(n, obj) – wektorowa funkcja bazowa $(\varphi_\nu(\underline{x}))$,

f(m,z) – skalarne funkcje ortonormalne $(f_\mu(z))$,

harmum[1.. dim] – tabela numerów dobieranych harmonik.

```
function fi(n: number; obj: object): real;
var
  m: number;
begin
```

```
        fi := 1; m := 1;
        while n > 0 do
          begin
            fi := fi * f(harmnum[m], obj[m]);
            m := m + 1; if m > dim then m := 1;
            n := n - harmnum[m];
          end
    end
```

6.3. Wybór liniowej funkcji przynależności

Mając wybraną ustaloną rodzinę funkcji Φ, stajemy przed zasadniczym problemem: w jaki sposób na podstawie ciągu uczącego U określić wartości współczynników wagowych V_η^i we wzorze (47)?

Zadanie to rozwiążemy w dwu etapach. Na początek rozważmy szczególną formę wzoru (47) przy założeniu, że $m = n$ oraz

$$\varphi_\nu(\underline{x}) = x_\nu, \tag{56}$$

przy czym z definicji

$$\varphi_o(\underline{x}) \equiv 1. \tag{57}$$

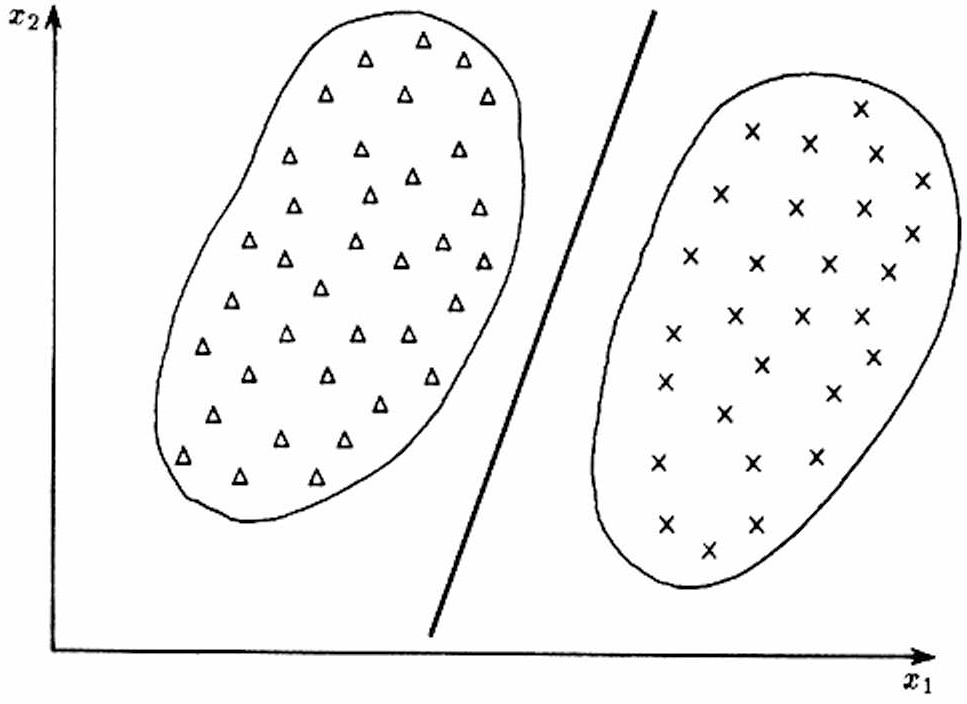
Wówczas oczywiście wzór (47) przyjmuje formę funkcji liniowej

$$C^i(\underline{x}) = \sum_{\nu=1}^{n} V_\eta^i x_\nu + V_0^i. \tag{58}$$

Przy takiej formie funkcji $C^i(\underline{x})$ reguły iteracyjnego określania współczynników wagowych V^i dają się wyjątkowo prosto zapisać, a następnie (co zostanie wykazane) wynik ten łatwo można uogólnić dla wzoru (47).

Warto przy tym zauważyć, że reguła rozpoznawania dana wzorem (58) nie ma bynajmniej wyłącznie teoretycznego znaczenia. Przeciwnie, jest to metoda ze wszech miar godna uwagi jako praktyczne narzędzie, warte zastosowania we wszystkich tych przypadkach, w których natura rozważanego problemu pozwala na zastosowanie takiej prostej reguły. Ponieważ

w praktyce prawie nigdy nie wiadomo, czy rozważany problem jest separowalny liniowo([1]) (rys 6.2), przeto celowe jest we wszystkich przypadkach podejmowanie próby wykorzystania metody (58) z gotowością do zaniechania jej w przypadku generacji zbyt dużej liczby błędów (rys 6.3). Argumentami przemawiającymi za celowością stosowania funkcji przynależności w postaci danej wzorem (58) są między innymi:



Rys. 6.2. Liniowa separowalność klas oznacza, że pomiędzy obszary należące do poszczególnych klas można wprowadzić granice w postaci hiperpłaszczyzny (na rysunku jest to linia prosta)

1) prostota algorytmu rozpoznawania (funkcję (58) łatwo zaprogramować i równie łatwo zrealizować w formie układu elektronicznego);

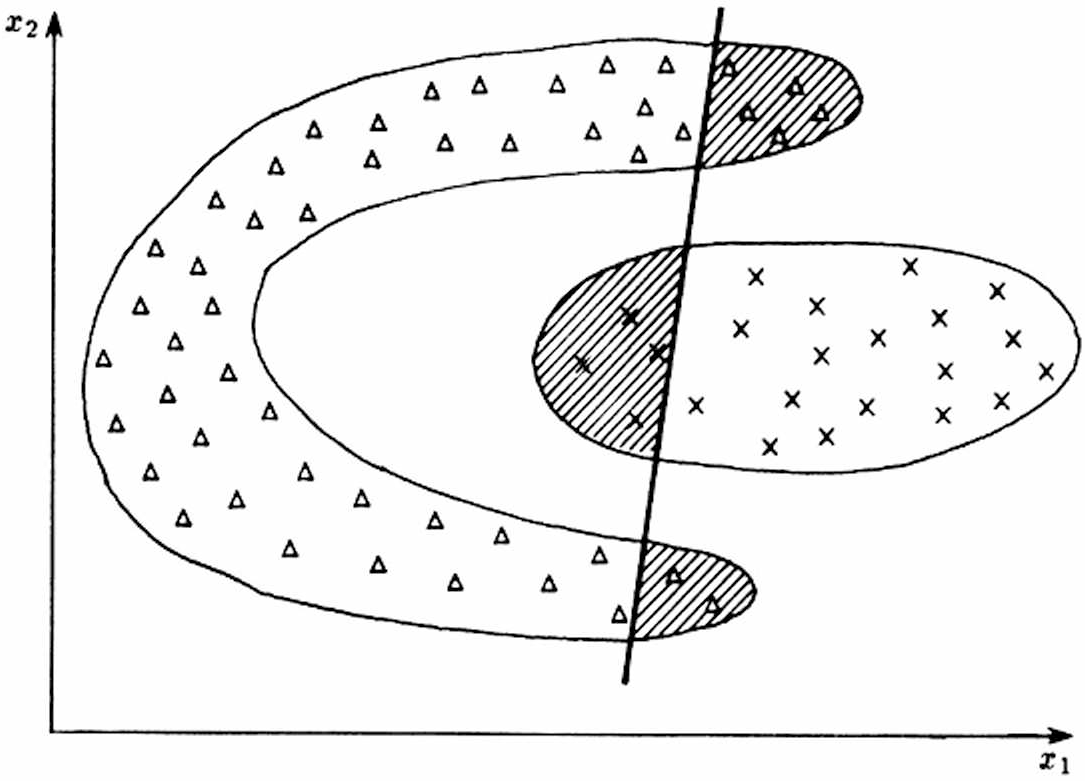
2) minimalna liczba koniecznych do pamiętania elementów (potrzeba jedynie $L(n+1)$ współczynników V_ν^i);

([1]) Związek pomiędzy kształtem funkcji przynależności a formą powierzchni rozgraniczającej obszary w przestrzeni X dany jest wzorem (50). Po wstawieniu do (50) funkcji danej formułą (58), otrzymuje się równanie hiperpowierzchni rozdzielającej, która jest płaszczyzną. Stąd stosowalność wzoru (58) jest ograniczona do zadań, w których obszary w przestrzeni cech mogą być rozgraniczane hiperpłaszczyznami (separowalne liniowo).

3) prosty w realizacji algorytm „uczenia" (ustalania na podstawie ciągu U wartości V^i) – patrz p. 6.4;

4) bezpośredni związek algorytmu z metodą NM;

5) prosty związek pomiędzy wzorem (58) a zasadami funkcjonowania mózgu człowieka [6].



Rys. 6.3. Przykład zadania, które nie jest liniowo separowalne: próba zastosowania hiperpłaszczyzny rozdzielającej spowoduje pojawienie się błędnych rozpoznań

Niektóre z przytoczonych argumentów wymagają bardziej szczegółowego rozwinięcia. Zacznijmy od argumentu 4, gdyż wcześniejsze albo są oczywiste, albo znajdą szersze uzasadnienie dalej. Istotnie, zakładając, że znane są wektory wzorców $\underline{M}^i$ oraz przyjmując metrykę Euklidesową (patrz D1.1), rozpoznajemy w metodzie NM obiekt $\underline{x}$ jako należący do klasy i, dla której odległość

$$\rho^2(\underline{x}, \underline{M}^i) = \sum_{\nu=1}^{n} (x - m_\nu^i)^2 \tag{59}$$

jest minimalna. Ale przekształcając wzór (59) łatwo otrzymać

$$\rho^2(\underline{x}, \underline{M}^i) = \sum_{\nu=1}^{n} x_\nu^2 - \left[\sum_{\nu=1}^{n} 2m_\nu^i x_\nu - \sum_{\nu=1}^{n}(m_\nu^i)^2\right]. \tag{60}$$

Pierwszy składnik wzoru (60) jest identyczny dla wszystkich klas i nie różnicuje odległości, zatem można go pominąć i rozpatrzyć jedynie fragment ujęty w nawiasy prostokątne, przy czym – ze względu na poprzedzający nawias znak minus – decyzja o przynależności obiektu będzie podejmowana na podstawie maksymalnej wartości fragmentu ujętego w nawias, który można wobec tego uznać za (inną niż zadana wzorem (40), ale prowadzącą do tych samych decyzji) funkcję przynależności dla metody NM. Zatem

$$C^i(\underline{x}) = \sum_{\nu=1}^{n} 2m_\nu^i x_\nu - \sum_{\nu=1}^{n}(m_\nu^i)^2 \tag{61}$$

i wystarczy dokonać kilku oczywistych podstawień, by utożsamić wzór (61) z formułą (58). Geometryczna interpretacja algorytmu funkcji liniowych (58) jest więc oczywista: opiera się ona na rozgraniczaniu obszarów *przynależnych* do określonego wzorca, przy czym – odmiennie niż w metodzie NM – twórca algorytmu nie musi się kłopotać wyszukiwaniem wzorców $\underline{M}^i$, gdyż odpowiednie obliczenia są dokonywane automatycznie w toku procesu uczenia (patrz p. 6.4).

Jeśli idzie o argument 5, to istotnie wzór (58) może być interpretowany jako najprostszy opis funkcjonowania neuronu (elementarnej komórki mózgu), zaś zadania rozpoznawania rozważane w praktyce bazują na możliwości rozpoznawania określonych obiektów przez człowieka. Do zagadnienia tego powrócimy w rozdziale 7.4.

Wymienione argumenty sprawiają, że metoda rozpoznawania oparta na wykorzystaniu wzoru (58), wprowadzona w książce jako etap rozważań nad własnościami ogólniejszej metody (47), jest przez wielu autorów traktowana jako niezależna, wartościowa metoda rozpoznawania. Naszkicujmy zatem – wzorem wcześniejszych rozdziałów – schemat algorytmu rozpoznawania dla tej metody. Algorytm ten oparty jest na identycznych założeniach, jak algorytmy prezentowane w poprzednich rozdziałach. Potrzebne jest jedynie wprowadzenie tablicy

weight [1 .. **numclass**][0 .. **dim**] – współczynniki wagowe (V_ν^i).

```
procedure linrec (obj, var rec);
begin
      for i := to numclass do
      begin
        fun[i] := weight[i][0];
        for n := 1 to dim do
          fun[i] := fun[i] + weight[i][n] * obj[n];
        end
      rec := pointmax(fun);
end
```

6.4. Metoda uczenia maszyny

Jak wynika z dotychczasowych rozważań, kluczowym problemem jest określenie wartości V_ν^i $(i = 1, \ldots, L;\ \nu = 0, 1, \ldots, n)$ na podstawie ciągu uczącego

$$U = \{\langle \underline{x}^k, i^k \rangle\}. \tag{62}$$

Przed zaproponowaniem odpowiedniej reguły uczenia wygodnie będzie wprowadzić kilka oznaczeń. Rozszerzmy wektor cech $\underline{x}$, wprowadzając składową o numerze 0 wartości (zawsze) wynoszącej 1; tak poszerzony wektor oznaczmy następująco:

$$\tilde{\underline{x}} = \langle x_0, x_1, x_2, \ldots, x_n \rangle \stackrel{\text{def}}{=} \langle 1, x_1, x_2, \ldots, x_n \rangle. \tag{63}$$

Oznaczmy ponadto wartość funkcji decyzyjnej F (por. (12)) dla obiektu $\underline{x}^k$ ciągu uczącego U przez

$$F^k \stackrel{\text{def}}{=} F\big(C^1(\underline{x}^k), C^2(\underline{x}^k), \ldots, C^l(\underline{x}^k)\big), \tag{64}$$

przy czym założymy, że

$$F^k \in I. \tag{65}$$

Wprowadzimy ponadto notację ułatwiającą śledzenie zmian wartości współczynników V_ν^i w czasie prezentacji kolejnych obiektów ciągu uczącego U. W tym celu wartości tych współczynników obowiązujące w k-tym kroku

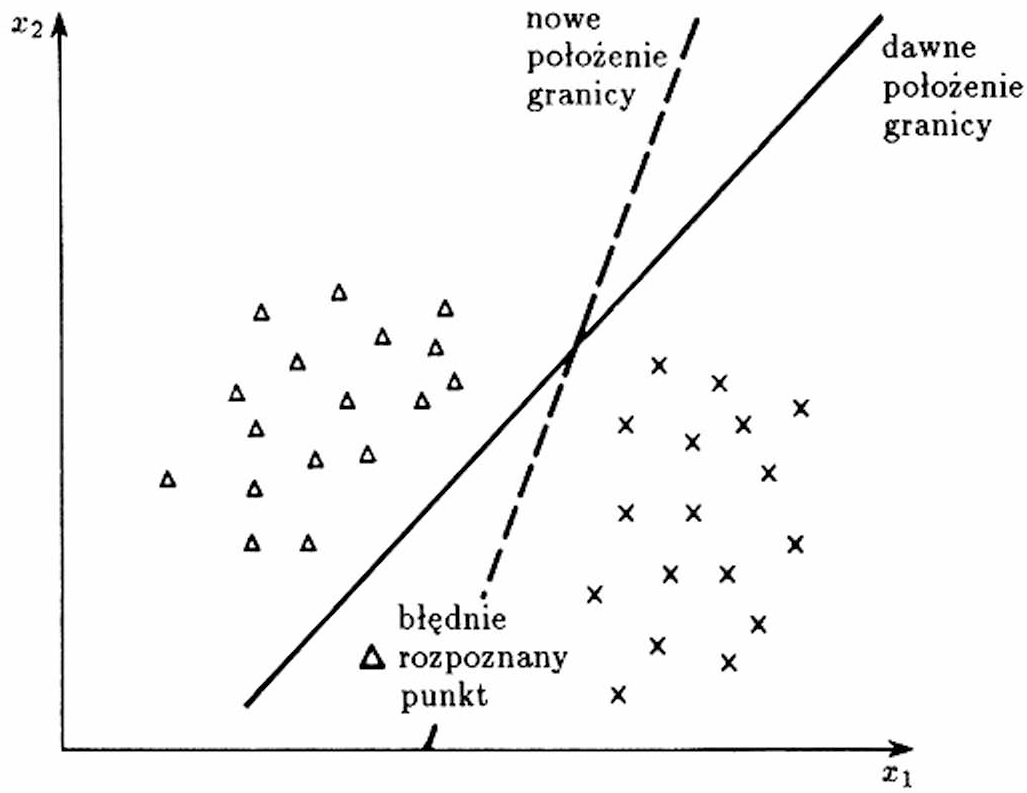
procesu uczenia podczas pokazu obiektu $\underline{x}^k$ o przynależności i^k oznaczymy $V_\nu^i(k)$.

W wyniku pokazu obiektu $\underline{x}^k$ obliczane są (na podstawie wartości $V_\nu^i(k)$) wszystkie funkcje przynależności $C^i(\underline{x}^k)$ i na ich podstawie podejmowana jest próba rozpoznawania, dająca w efekcie (zgodnie z przyjętą konwencją) decyzję F^k. Reguła uczenia, pozwalająca ulepszyć zestaw wag $V_\nu^i(k)$ na podstawie znajomości $\underline{x}^k$, i^k oraz F^k, jest następująca:

$$V_\nu^{i^k}(k+1) = V_\nu^{i^k}(k) + \tilde{x}_\nu^k, \tag{66}$$

$$V_\nu^{F^k}(k+1) = V_\nu^{F^k}(k) - \tilde{x}_\nu^k, \tag{67}$$

przy czym $\nu = 0, 1, 2, \ldots, n$. Metoda ta ma prostą interpretację geometryczną (rys. 6.4).

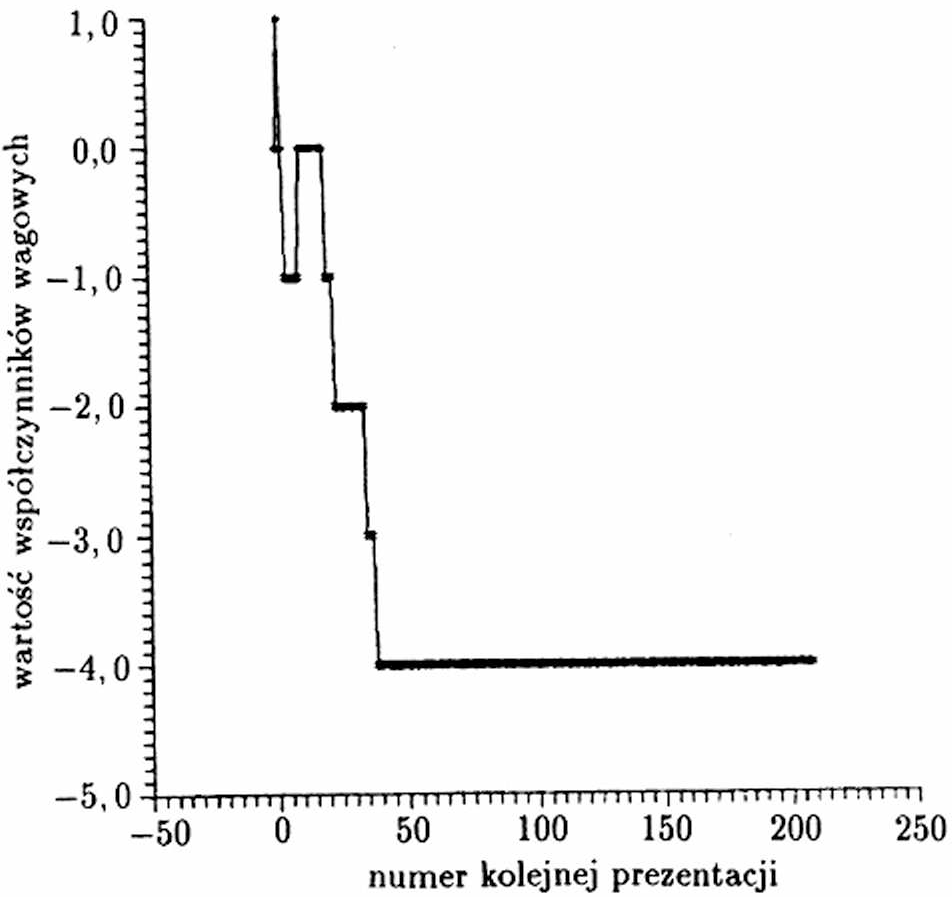


Rys. 6.4. Proces uczenia polega na takim przemieszczaniu granicznej płaszczyzny, aby punkt błędnie klasyfikowany przy aktualnych wartościach współczynników wagowych był po korekcie klasyfikowany poprawnie

Należy zauważyć, że w procesie uczenia ulegają zmianie jedynie wagi funkcji, które albo powinny były ($i = i^k$), albo w rzeczywistości ($i = F^k$) przyjęły wartości maksymalne; pozostałe $V^i(k)$ pozostają bez zmian dla kroku $(k+1)$

$$V_\nu^i(k+1) = V_\nu^i(k), \quad i \neq i^k \quad i \neq i^F \quad \nu = 0, \ldots, n. \tag{68}$$

W przypadku kiedy próba rozpoznawania obiektu powiedzie się (co oznacza, że $F^k = i^k$), wówczas żaden ze współczynników wagowych nie ulega zmianie (gdyż korekty zadawane wzorami (66) i (67) znoszą się).

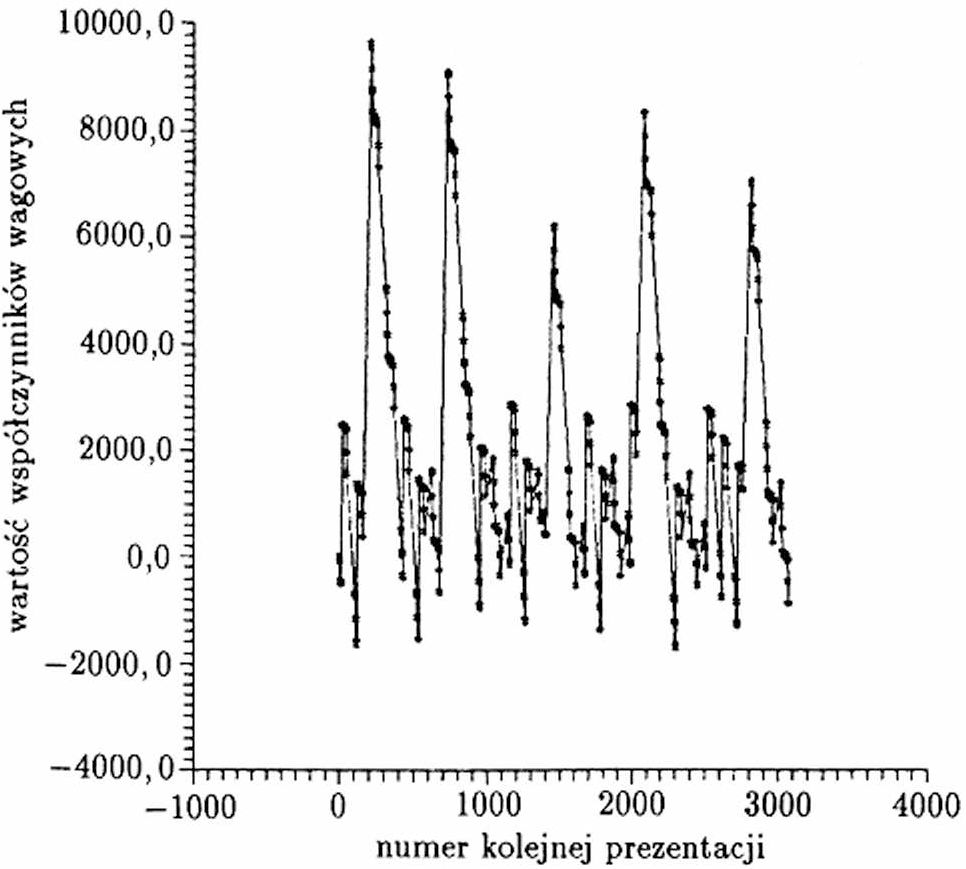


Rys. 6.5. Jeśli obszary rozpoznawanych klas są liniowo separowalne, to kolejne wartości współczynników wagowych uzyskiwane w czasie uczenia dążą asymptotycznie do pewnych ustalonych wartości (rysunek pochodzi z pracy doktorskiej Anny Popieli-Mizery)

W ten sposób po znalezieniu zestawu wag V_ν^i gwarantujących bezbłędne funkcjonowanie algorytmu proces uczenia *de facto* zatrzyma się.

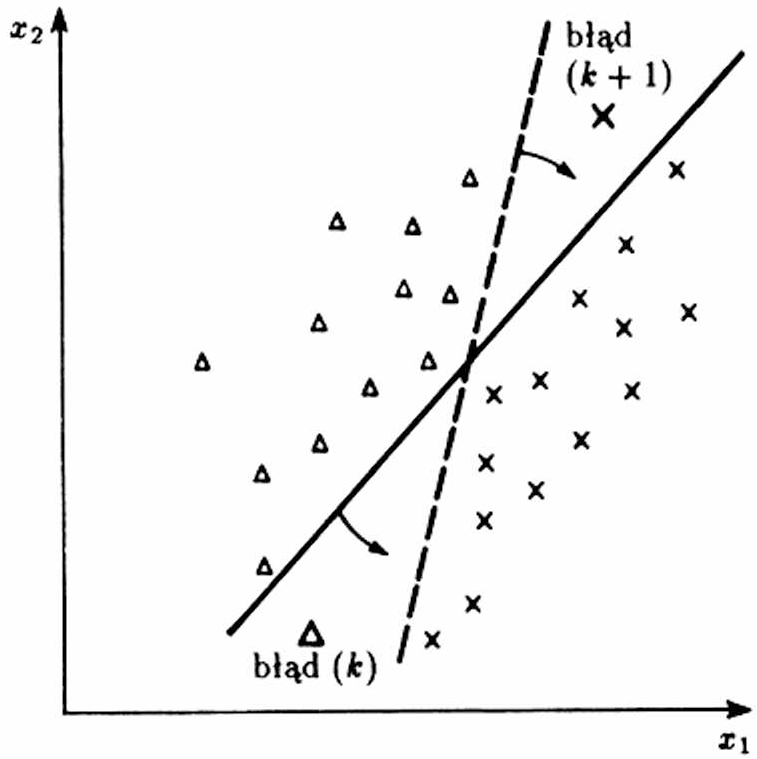
Można udowodnić (dowód, ze względu na jego znaczenie, przytoczony jest – wyjątkowo – w Dodatku 2), że niezależnie od sposobu wybrania początkowych wartości współczynników wagowych $V_\nu^i(1)$, *po skończonej liczbie pokazów* obiektów ciągu uczącego $\{\langle \underline{x}^k, i^k \rangle\}$ nastąpi ustalenie wartości V_ν^i, pozwalające bezbłędnie klasyfikować wszystkie obiekty([2]).

Naturalnie teza przytoczonego twierdzenia funkcjonuje jedynie wtedy,



Rys. 6.6. Brak możliwości liniowej sepracji klas objawia się tym, że wartości współczynników wagowych wykazują niegasnące oscylacje (rysunek pochodzi z pracy doktorskiej Anny Popieli-Mizery)

([2]) Formalnie poprawne rozpoznawanie jest zapewnione jedynie dla obiektów ciągu uczącego, ale zakładając jego reprezentatywność możemy przypuszczać, że gwarancja obejmuje wszystkie obiekty.



Rys. 6.7. Poprawka położenia linii granicznej spowodowana przez jeden błędnie sklasyfikowany punkt (gwiazdka) wywoła błędne sklasyfikowanie już uprzednio poprawnie rozpoznawanych punktów (trójkąt) i kolejne korekty w przeciwną stronę

gdy rozdzielenie za pomocą hiperpłaszczyzn jest w ogóle wykonalne lub – formułując to samo inaczej – gdy istnieją funkcje postaci (58), pozwalające na bezbłędne rozpoznawanie. Mimo tego ograniczenia teza twierdzenia jest bardzo mocna; głosi ona, że stosując wzory (66) i (67) zawsze znajdziemy rozwiązanie i to po skończonej liczbie prezentacji obiektów ciągu uczącego (który w razie potrzeby – gdyby niezbędna liczba pokazów okazała się większa od N – może być prezentowany cyklicznie).

Pokażmy teraz strukturę algorytmu uczenia. Musimy jednak do wcześniej wprowadzonych informacji i założeń dodać kilka dodatkowych zmiennych:

count – licznik powtórzeń prezentacji ciągu uczącego,

limit – maksymalna liczba pokazów, po której przerywa się cykliczne przeszukiwanie ciągu uczącego sygnalizując niepowodzenie procesu uczenia,

error – zmienna logiczna ustawiana w momencie stwierdzenia pomyłki przy próbie rozpoznawania,

linrec(sampl[k]) – funkcja dostarczająca numeru klasy rozpoznanej dla aktualnie rozważanego obiektu przez algorytm funkcji liniowych (może to być algorytm wprowadzony w poprzednim podrozdziale),

fail – procedura wywoływana w przypadku niepowodzenia,

success – procedura wywoływana w wyniku nauczenia procedury poprawnego rozpoznawania.

Zakładamy też, że tablica **sampl** (zawierająca ciąg uczący) została rozszerzona w ten sposób, że zawiera także zerowe składowe wektorów $\tilde{\underline{x}}$ (wszystkie równe 1). Tablica ma wymiary sampl[1 .. num][0 .. dim + 1].

```
procedure learnig;
begin
      weight := 0;      {wyzerowanie całej tablicy}
      count := 0;
      repeat
        error := FALSE;
        for k := 1 to num do
        begin
          rec := linrec(sampl[k]);
          ident := sampl[k][dim + 1];
          error := rec <> ident;
          if error then
            for n := 0 to dim do
            begin
              weight[ident][n] := weight[ident][n] + sampl[k][n];
              weight[rec][n] := weight[rec][n] - sampl[k][n];
            end;
        end
        count := count + 1;
        until not error or count > limit;
      if error then fail else success;
end
```

Warto zwrócić uwagę na fakt, że zbieżność procesu uczenia można ocenić bezpośrednio z obserwacji przebiegu zmian wartości $V_\nu^i(k)$ w funkcji kolejnych pokazów k. Jeśli klasy są liniowo rozdzielne i proces uczenia jest zbieżny, to wówczas wartości wag asymptotycznie dążą do swych wartości ustalonych (rys. 6.5), w przeciwnym razie obserwuje się silne oscylacje

(rys. 6.6), pochodzące od ciągłych obrotów płaszczyzny granicznej, wymuszanych błędnymi rozpoznaniami (rys. 6.7).

6.5. Metoda funkcji nieliniowych

Możemy teraz wrócić do ogólnego sformułowania zadania (wzór (47)). Jakie zalety może mieć zastosowanie ogólnej formuły (47) w stosunku do wzoru dla funkcji liniowych (58)? Otóż wydaje się, że można wskazać przynajmniej dwa powody, dla których użycie formuły (47) może przynieść sukces w warunkach, kiedy wzór (58) nie doprowadził do zadowalających wyników:

1. Stosując wzór (47), możemy generować w przestrzeni cech powierzchnie rozgraniczające o dowolnych kształtach (por. (50)), a nie tylko hiperpłaszczyzny. Pozwala to na rozgraniczenie obszarów liniowo nieseparowalnych. Już tylko zastosowanie we wzorze (47) wielomianów drugiego stopnia pozwala uzyskiwać jako powierzchnie separujące: elipsoidy, paraboloidy, hiperboloidy oraz ich dowolne kombinacje.

2. Stosując wzór (47), mamy do dyspozycji m dobieranych współczynników, przy czym na ogół $m \gg (n+1)$. Istnieje zaś twierdzenie pokazujące związek pomiędzy liczbą nastawialnych wag a prawdopodobieństwem poprawnego rozpoznawania (por. [7]), przy czym wniosek z tego twierdzenia wskazuje na celowość zwiększania m w przypadku trudności z rozpoznawaniem.

Tak więc celowe jest przyjęcie metody rozpoznawania opartej na funkcji przynależności postaci:

$$C^i(\underline{x}) = \sum_{\nu=0}^{m} V_\nu^i \varphi_\nu(\underline{x}). \tag{69}$$

Jak w przypadku takiej funkcji prowadzić proces uczenia, to znaczy znajdowania wag V_ν^i ? Otóż łatwo wykazać, że skuteczny jest ponownie algorytm dany wzorami (66) i (67) pod warunkiem odmiennego zinterpretowania zmodyfikowanej zmiennej $\underline{x}$. Załóżmy, że związek pomiędzy wektorem cech $\underline{x}$ a wektorem zmodyfikowanym $\tilde{\underline{x}}$ jest dany nie wzorem (63), lecz zależnością:

$$\tilde{x}_\nu = \varphi_\nu(\underline{x}); \qquad \nu = 0, 1, \ldots, m. \tag{70}$$

Wówczas

$$C^i(\underline{x}) = C^i(\tilde{\underline{x}}) = \sum_{\nu=0}^{m} V_\nu^i \tilde{x}_\nu \tag{71}$$

i wzory (66), (67) i (68) mogą być stosowane bez ograniczeń. Możemy to interpretować między innymi w ten sposób, że wzór (70) zadaje transformację n-wymiarowej przestrzeni cech w m-wymiarową przestrzeń „prostującą", w której to przestrzeni obszary należące do różych klas, nieseparowalne liniowo w przestrzeni cech, dają się rozgraniczać hiperpłaszczyznami (71).

Ocena szans powodzenia rozpoznawania w konkretnym zadaniu z wykorzystaniem metod funkcji przynależności $C^i(\underline{x})$ mających postać agregatów określonych funkcji φ_ν jest bardzo trudna. Jednak w sposób dość ogólny można wykazać, że prawdopodobieństwo poprawnego rozpoznania wyraźnie wzrasta przy przejściu od funkcji $C^i(\underline{x})$ liniowych do – bogatszych w możliwości – funkcji nieliniowych. Dzieje się tak dlatego, że – jak wspomniano – na ogół $m \gg n$. Na przykład dla funkcji $C^i(\underline{x})$ kwadratowej $m = n(n+3)/2$, zaś dla funkcji $C^i(\underline{x})$ będącej wielomianem stopnia μ

$$m = \binom{n+\mu}{\mu} - 1,$$

gdzie

$$\binom{n+\mu}{\mu} = \frac{(n+\mu)!}{n!\mu!}$$

oznacza symbol Newtona.

Wpływ wielkości m na prawdopodobieństwo prawidłowego rozpoznania oszacować można na podstawie następującego prostego rozumowania: N punktów (na przykład wszystkie punkty ciągu uczącego) podzielić można na dwie klasy (rozważamy najprostszy przypadek $L = 2$) ogółem na 2^N sposobów([3]). Za pomocą hiperpłaszczyzn([4]) w m-wymiarowej przestrzeni te N punktów podzielić można na $L(N, m)$ sposobów. Zatem prawdopodobieństwo tego, że uda się za pomocą generowanego w procesie uczenia

([3]) Zakładamy, że wszystkie punkty znajdują się w m-wymiarowej przestrzeni w tak zwanym ogólnym położeniu, co oznacza, że żaden podzbiór $m+1$ punktów nie leży w całości na żadnej $(m-1)$-wymiarowej hiperpłaszczyźnie (na przykład dla $m = 2$ oznacza to, że żadne trzy punkty nie leżą na jednej prostej).

([4]) Funkcje $C^i(\underline{x})$ generują zawsze podziały liniowe m-wymiarowej przestrzeni prostującej, zadanej transformacją $\tilde{x}_\nu = \varphi_\nu(\underline{x})$.

liniowego podziału odtworzyć potrzebne przyporządkowanie N punktów do dwóch klas szacować można za pomocą formuły:

$$Prawd(N,m) = \frac{L(N,m)}{2^N}.$$

W zasadzie dla uzasadnienia tezy, że zwiększanie m zwiększa szansę sukcesu przy rozpoznawaniu wystarczyło by wykazać, że $Prawd(N,m)$ jest rosnącą funkcją swego drugiego argumentu. Jednak specyfika tego wzrostu skłania do pokazania tu kilku szczegółów. W tym celu musimy bliżej określić funkcję $L(N,m)$. Stosunkowo łatwo jest określić jej wartości dla małych N oraz $m = 2$, na przykład z rysunku 6.8 łatwo wywnioskować([5]), że $L(4,2) = 14$. Dowolne wartości $L(N,m)$ mogą być obliczone z formuły rekurencyjnej:

$$L(N,m) = L(N-1,m) + L(N-1,m-1),$$

wykorzystywanej waraz z oczywistymi warunkami granicznymi:

$$L(1,m) = 2 \quad \text{oraz} \quad L(N,1) = 2N;$$

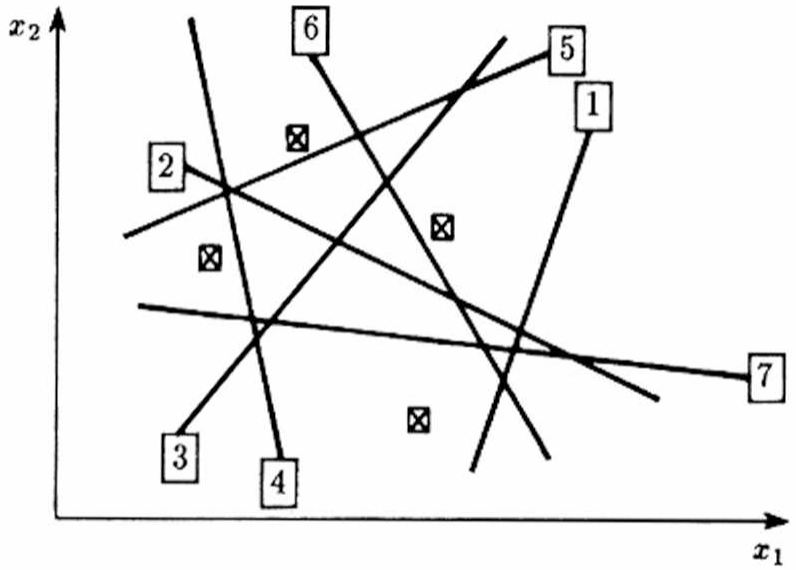
ewentualnie można je otrzymywać wprost ze wzoru:

$$L(N,m) = \begin{cases} 2\sum_{\mu=0}^{n} \binom{N-1}{\mu} & \text{dla } N > m, \\ 2^N & \text{dla } N \leq m. \end{cases}$$

Dość uciążliwa obliczeniowo postać obydwu zależności uzasadnia celowść pokazania zmienności tej funkcji w postaci tabelarycznej (zebrano wartości $L(N,m)$ dla $N < 8$ i $m < 6$) – tabela 6.1

Znając wartości funkcji $L(N,m)$ możemy określić przebieg prawdopodobieństwa $Prawd(N,m)$. Łatwo zauważyć, że $\lim_{m\to 0} Prawd(N,m) = 0$ oraz, zgodnie z zapowiedzią, $\lim_{m\to\infty} Prawd(N,m) = 1$. Przydatne z punktu

([5]) Analizując rysunek warto zwrócić uwagę, że każda z narysowanych linii (będących szczególnymi przypadkami hiperpłaszczyzny dla $m = 2$) generuje *dwa* możliwe podziały: przykładowo jeden, w którym punkty x_1 i x_2 należą do klasy $i = 1$, zaś punkty x_3 i x_4 do klasy $i = 2$, oraz drugi, w którym x_1 i x_2 należą do klasy $i = 2$, zaś x_3 i x_4 należą do klasy $i = 1$.



Rys. 6.8. Dychotomie liniowe dla $m = 2$ oraz $N = 4$

Tabela 6.1. Wartość funkcji $L(N, m)$ dla $N < 8$ i $m < 6$

Liczba punktów	Wymiar przestrzeni cech				
	1	2	3	4	5
1	2	2	2	2	2
2	4	4	4	4	4
3	6	8	8	8	8
4	8	14	16	16	16
5	10	22	30	32	32
6	12	32	52	62	64
7	14	44	84	114	126
8	16	58	128	198	240

widzenia zastosowań jest natomiast stwierdzenie, że $Prawd(N, m^*) = 1/2$ dla $m^* = \frac{1}{2}N - 1$, a także dostrzeżenie faktu, że w pobliżu m^* następuje szybkie przejście od $Prawd(N, m) = 0$ do $Prawd(N, m) = 1$. Przejście to jest szczególnie gwałtowne dla dużych N, można bowiem wykazać, że

$$\lim_{N\to\infty} Prawd(N, m^* + \varepsilon) = 1,$$

$$\lim_{N\to\infty} Prawd(N, m^* - \varepsilon) = 0,$$

dla każdego (dowolnie małego) ε. Jak z tego wynika, początkowy brak powodzenia przy próbach rozpoznawania można (z prawdopodobieństwem zmierzającym do jedności!) zamienić na sukces zwiększając m. Oczywiście zwiększanie wymiaru przestrzeni m musi odbywać się poprzez wprowadzanie takich nowych składowych, które istotnie wnoszą nowe informacje. Mogą to być dodatkowe cechy x_i, ale mogą to być także nowe transformacje([6]) φ_ν już zarejestrowanych cech $\underline{x}$. Niezależnie jednak od tego, jakie mechanizmy zwiększania m zastosujemy, należy wykazać cierpliwość i konsekwencję, ponieważ nigdy nie wiadomo, jak daleko jest do wartości krytycznej m^*.

W związku z prostą i naturalną relacją, jaka istnieje pomiędzy omówioną metodą rozpoznawania z wykorzystaniem funkcji liniowych (p. 6.3) a stosowaniem tu omówionej, ogólniejszej metody, bezcelowe jest oddzielne formułowanie algorytmu dla tego przypadku, ponieważ uogólnienie poprzedniego algorytmu sprowadza się wyłącznie do wbudowania weń funkcji odpowiedzialnej za dokonywanie transformacji $\tilde{x}_\nu = \varphi_\nu(\underline{x})$ (por. także p. 6.2) oraz do przewymiarowania odpowiednich tablic (zgodnie z faktem, że $m \gg (n+1)$). Również procedura uczenia nie wymaga odrębnego omówienia, gdyż jest praktycznie identyczna z omówioną (p. 6.4). Jest ona zresztą dodatkowo omówiona i poparta algorytmem w następnym rozdziale.

([6]) Poprzez transformacje ujawniane są nowe właściwości używanych cech, dlatego proces ten można także rozpatrywać jako formę wzbogacenia wiedzy o rozpoznawanych obiektach.

7. METODY SPECJALNE

7.1. Podstawowe sformułowanie metody funkcji potencjalnych

W licznych pracach teoretycznych (por. np. [5]), a także w wielu zastosowaniach rekomendowana jest metoda funkcji potencjalnych jako jedna z bardziej skutecznych metod rozpoznawania obrazów. W istocie metoda ta ma własności zbliżone do własności omówionych wcześniej metod minimalnoodległościowych i aproksymacyjnych, a jej głównymi zaletami są intuicyjnie zrozumiała zasada i starannie udokumentowana matematycznie poprawność. (Odpowiednie rozważania, twierdzenia i dowody matematyczne znaleźć można w książce [5]). Koncepcja metody funkcji potencjalnych polega na tym, aby funkcje przynależności $C^i(\underline{x})$ budować jako superpozycje funkcji([1]) $K(\underline{x}, \underline{x}^{i,k})$ przypominających swoim kształtem rozkład potencjału elektrycznego wokół ładunku punktowego (stąd nazwa metody – por. rys. 7.1). Funkcje składowe (*potencjalne*) są silnie malejącymi funkcjami odległości pomiędzy punktem generującym *potencjał* $\underline{x}^{i,k}$ (należącym do ciągu uczącego U^i) a punktem $\underline{x}$, w którym obliczana jest wartość funkcji potencjalnej:

$$\lim_{\rho(\underline{x},\underline{x}^{i,k})\to\infty} K(\underline{x}, \underline{x}^{i,k}) = 0.$$

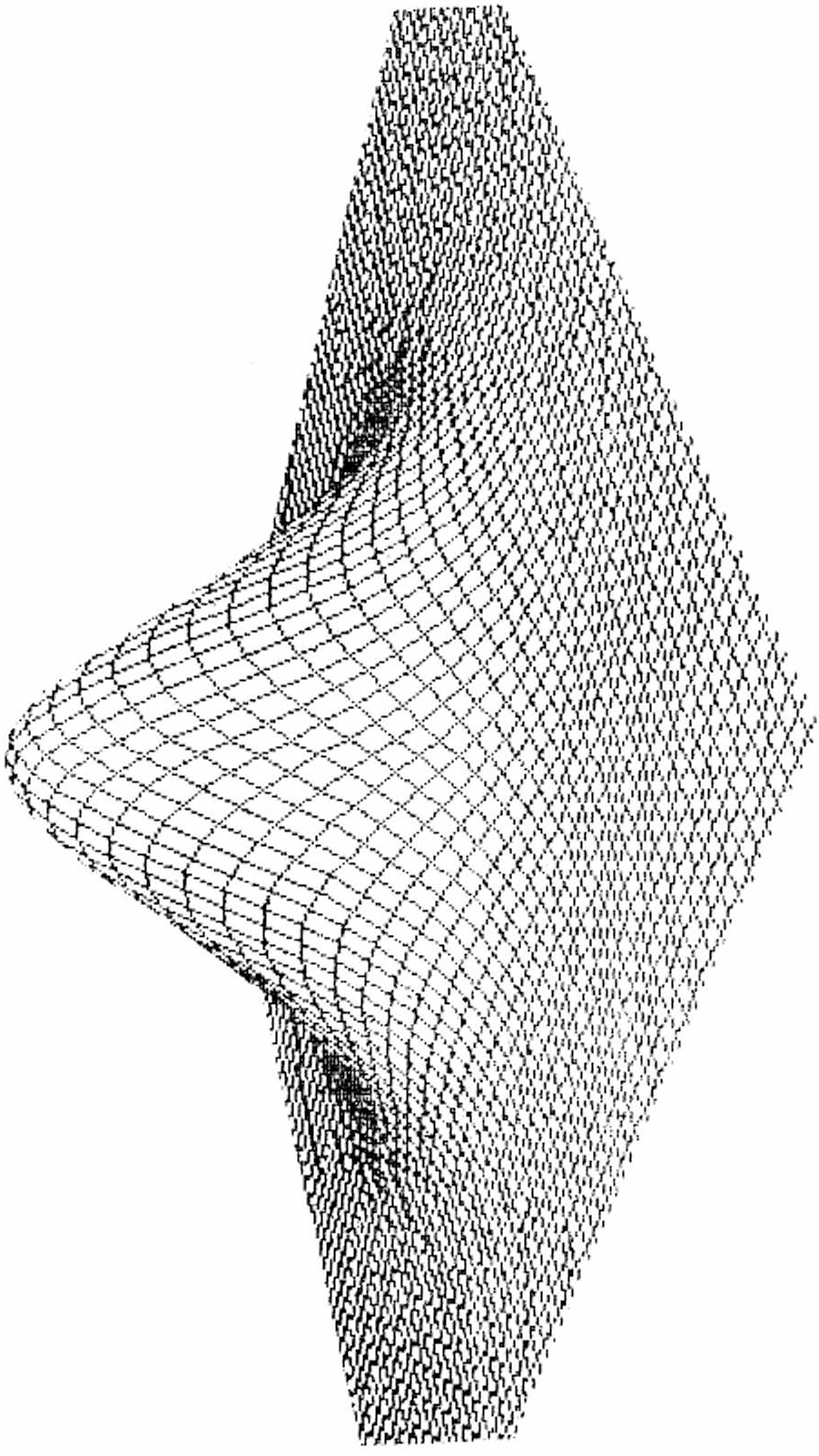
Przykład. W literaturze rozważa się między innymi funkcje:

$$K(\underline{x}, \underline{x}^{i,k}) = \exp\left(-\mu\rho^2(\underline{x}, \underline{x}^{i,k})\right) \tag{72}$$

lub

$$K(\underline{x}, \underline{x}^{i,k}) = \frac{1}{1 + \mu\rho^2(\underline{x}, \underline{x}^{i,k})}, \tag{73}$$

([1]) Funkcje potencjalne oznaczano zwykle jako $K(\underline{x}, \underline{x}^{i,k})$, pomimo że w istocie były to funkcje jednego argumentu wektorowego $\underline{x}$, zaś $\underline{x}^{i,k}$ był parametrem.



Rys. 7.1. Przykładowy kształt funkcji potencjalnej

przy stałej $\mu > 0$ i dowolnie wybranej metryce ρ (por. Dodatek 1).

Mając wybraną funkcję potencjalną, można zapisać funkcje przynależności w postaci sum

$$C^i(\underline{x}) = \sum_{k=1}^{N_i} \eta_k K(\underline{x}, \underline{x}^{i,k}), \tag{74}$$

gdzie η_k oznacza elementy ciągu liczbowego mającego charakter uzbieżniający. W wielu rekomendacjach praktycznych ciąg ten jest pomijany ($\eta_k = 1$), lecz z wywodów teoretycznych wynika [5], że celowe jest stosowanie ciągu η_k o własnościach

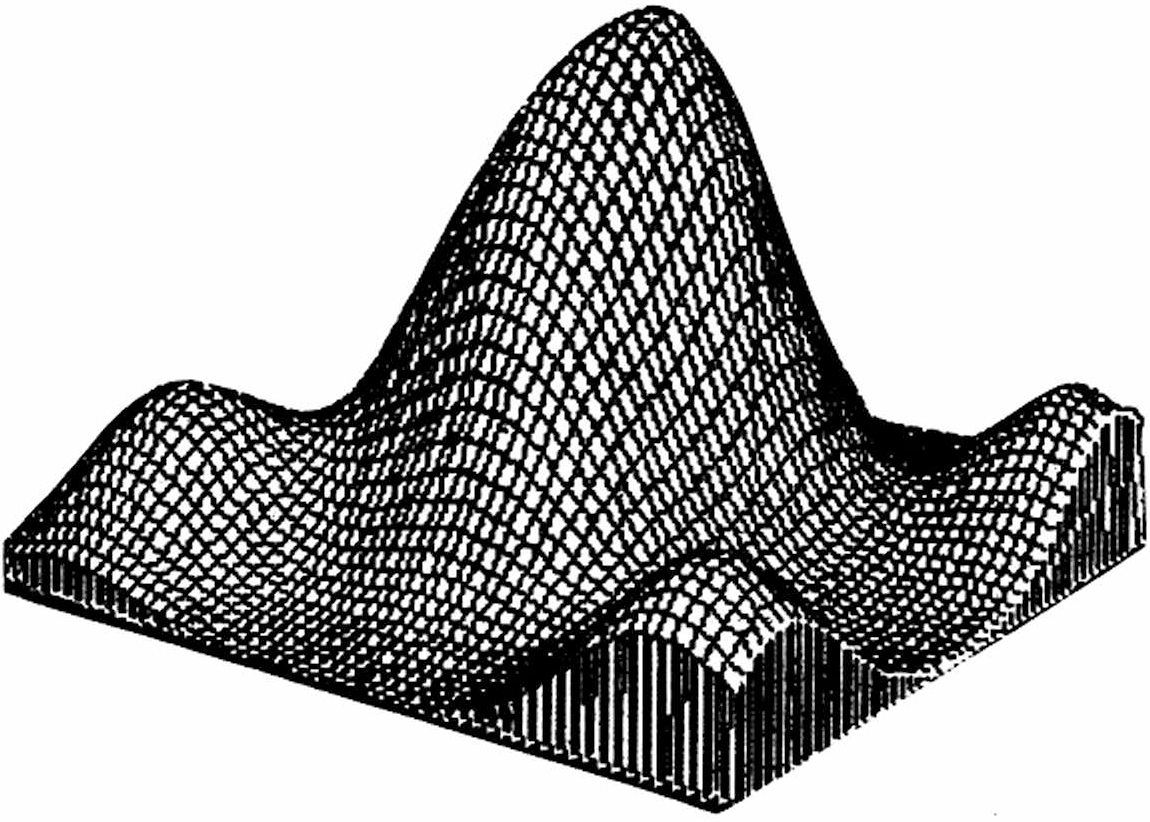
$$\sum_{k=1} \eta_k = \infty \tag{75}$$

oraz

$$\sum_{k=1} \eta_k^2 < \infty. \tag{76}$$

Przykład. Wzór (74) proponuje bardzo sugestywną intuicyjnie metodę rozpoznawania: każdy punkt ciągu uczącego wytwarza w przestrzeni X strefę o zwiększonej wiarygodności napotkania obiektów należących do klasy, do której należał rozważany punkt ciągu uczącego. Wpływ tego punktu maleje jednak szybko ze wzrostem odległości, a funkcja przynależności do określonej klasy jest superpozycją stref pochodzących od poszczególnych obiektów danej klasy (rys. 7.2).

Dla osób z wyobraźnią przestrzenną obraz funkcji przynależności jako superpozycji *dzwonów* postaci (72) ma ogromny urok. Na tym jednak zalety się kończą. Popatrzmy uważnie na wzór (74). Do jego stosowania *konieczne jest pamiętanie wszystkich wartości* $\underline{x}^{i,k}$, czyli w efekcie całego ciągu uczącego U! Metoda ma więc tę samą własność, ktora przesądzała o małej przydatności metody NN i pochodnych, a jednocześnie wzór (74) wymaga bardziej złożonych i czasochłonnych obliczeń niż wzór (23). Autorzy metody funkcji potencjalnych dostrzegli tę wadę i zaproponowali *perceptronową* realizację metody. Zanim jednak ją przedstawimy, trzeba podsumować dotychczasowe rozważania w postaci odpowiedniego algorymu. Do jego wprowadzenia konieczne jest uzupełnienie definicji wykorzystywanych funkcji:



Rys. 7.2. Funkcja przynależności jako superpozycja powierzchni odpowiadających funkcjom potencjalnym generowanym przez obiekty ciągu uczącego

fpot(obj, sampl[k]) – funkcja potencjalna $\left(K(\underline{x}, \underline{x}^{i,k})\right)$,

seq(k) – funkcja generująca ciąg uzbieżniający (η_k).

```
procedure fpotrec1 (obj, var rec);
begin
      fun = 0;      {wyzerowanie całej tablicy}
      for k := 1 to nym do
        begin
          ik := sampl[k][dim+1];
          fun[ik] := fun[ik] + seq(k) * fpot(obj, sampl[k]);
        end
      rec := pointmax(fun);
end
```

7.2. Metoda funkcji potencjalnych w realizacji perceptronowej

Załóżmy, że funkcję potencjalną można przedstawić w postaci:

$$K(\mu, \eta) = \sum_{\nu=0}^{m} \varphi_\nu(\mu)\varphi_\nu(\eta). \tag{77}$$

Wówczas łatwo wykazać, że ze wzoru (74) otrzymujemy funkcję:

$$C^i(\underline{x}) = \sum_{\nu=0}^{m} V_\nu^i \varphi_\nu(\underline{x}) \tag{78}$$

oraz regułę obliczania wartości wag V_ν^i

$$V_\nu^{i^k}(k+1) = V_\nu^{i^k}(k) + \eta_k \varphi_\nu(\underline{x}^k), \tag{79}$$

$$V_\nu^{F^k}(k+1) = V_\nu^{F^k}(k) - \eta_k \varphi_\nu(\underline{x}^k), \tag{80}$$

czyli (z dokładnością do ciągu η_k) są to reguły wprowadzone w poprzednim rozdziale([2]). Tak więc metoda funkcji potencjalnych jest w istocie identyczna z metodami już poznanymi, co jest stwierdzeniem istotnym z powodu licznych literaturowych powołań na tę metodę([3]).

Zapiszmy jeszcze algorytm dla opisanej realizacji metody funkcji potencjalnych, wykorzystując w nim dodatkowo:

limit – liczba funkcji $\varphi_\nu(\underline{x})$ używanych w rozwinięciu (**m**),

fi(m,obj) – funkcje bazowej rodziny $\varphi_\nu(\underline{x})$.

Algorytm opiszemy, wyróżniając w nim procedurę **learning** (uczenie) i funkcję **recognition** (rozpoznawanie).

([2]) W celu uwypuklenia identyczności metody funkcji potencjalnych z metodami aproksymacyjnymi zmieniono całkowicie stosowaną w teorii funkcji potencjalnych notację, uzgadniając ją z oznaczeniami stosowanymi w książce.

([3]) Fakt zawierania się licznych metod heurystycznych w ogólnym schemacie metody funkcji potencjalnych, traktować można jako zaletę metody, gdyż stwarza ona ogólniejszą podstawę do budowy teorii rozpoznawania.

```
procedure learning;
{tu powinny być zadeklarowane wszystkie stałe, zmienne i tablice}
function recognition (obj: object): class;
var
  i, m: integer;
begin
      for i := 1 to numclass do
        begin
          fun[i] := 0;
          for m := 0 to limit do
            fun[i] := fun[i] + weight[i][m] * fi(m, obj);
        end
      recognition := pointmax(fun);
end; {koniec funkcji recognition}

begin{początek procedury learning}
      weight := 0; {wyzerowanie całej tablicy}
      count := 0;
      repeat
        error := FALSE;
        for k := 1 to num do
        begin
          rec := recognition(sampl[k]);
          ident := sampl[k][dim + 1];
          error := rec <> ident;
          if error then
            for n := 0 to limit do
            begin
              weight[ident][n] := weight[ident][n] + seq(k) * fi(n, sampl[k]
              weight[rec][n] := weight[rec][n] - seq(k) * fi(n, sampl[k][n]);
            end;
        end
        count := count + 1;
      until not error or count > limit;
      if error then fail else success;
end
```

7.3. Metoda aproksymacji stochastycznej

Stosunkowo często w literaturze (zwłaszcza zachodniej) rekomendowane są techniki rozpoznawania bazujące na metodzie aproksymacji stochastycznej Robbinsa i Monro [8]. Jej zaletą jest solidne oparcie metody na teorii procesów stochastycznych, co tłumaczy jej dużą popularność. Wprawdzie w książce [5] udowodniono, że metoda funkcji potencjalnych jest równoważna pod każdym względem metodzie aproksymacji stochastycznej, zaś metoda funkcji potencjalnych została opisana, jednak ze względu na popularność tego ujęcia i jego przydatność w pewnym uproszczonym wariancie zadania – można uznać, że opisanie tu techniki aproksymacji stochastycznej jest celowe.

Metoda Robbinsa i Monro zdefiniowana została w celu wyznaczania miejsc zerowych (pierwiastków) równania regresji o ogólnej postaci:

$$\mathsf{E}_{\underline{x}}\{\Phi(V_1, V_2, V_3, \ldots, V_m, \underline{x})\} = 0,$$

gdzie Φ jest pewną funkcją argumentu wektorowego $\underline{x}$, o którym zakłada się, że jest realizacją pewnego procesu stochastycznego, a E oznacza wartość oczekiwaną. Aby tę metodę wykorzystać do rozpoznawania, trzeba utożsamić funkcję Φ z jakimś pojęciem z zakresu rozpoznawania obrazów. Zazwyczaj robi się to wprowadzając pojęcie funkcji rozdzielającej. Funkcja taka może wskazywać na właściwą klasę $i \in I$ poprzez swój znak. Istotnie, w zagadnieniach *dychotomii* (rozpoznania jednej z dwóch rozpatrywanych klas) można posłużyć się funkcją tego rodzaju. Załóżmy bowiem, że rozpoznawane są wyłącznie dwie klasy o numerach $i_1 \in I$ oraz $i_2 \in I$ z takim założeniem, że każdy obiekt $d \in D$ musi należeć do jednej z wymienionych klas:

$$\forall_{d \in D} A(d) \neq i_1 \Rightarrow A(d) = i_2.$$

Wówczas zamiast funkcji przynależności $C^i(\underline{x})$ można posłużyć się funkcją rozdzielającą o postaci:

$$C(\underline{x}) = C^{i_1}(\underline{x}) - C^{i_2}(\underline{x}).$$

Funkcja ta decyduje o przynależności obiektów na podstawie swego znaku:

$$F(C(\underline{x})) = \begin{cases} i_1 & \text{dla} \quad C(\underline{x}) > 0, \\ i_2 & \text{dla} \quad C(\underline{x}) < 0. \end{cases}$$

Łatwo zauważyć, że funkcja rozdzielająca opisuje granicę pomiędzy obszarami klas i_1 oraz i_2 za pomocą równania $C(\underline{x}) = 0$, które spełnia założenia metody aproksymacji stochastycznej, przy podstawieniu:

$$\Phi(V_1, V_2, V_3, \ldots, V_m, \underline{x}) = C(\underline{x}) = \sum_{\nu=0}^{m} V_\nu \varphi_\nu(\underline{x}).$$

Można więc zastosować iteracyjną procedurę Robbinsa i Monro do ustalenia wartości wag V_μ:

$$V_\mu(k+1) = V_\mu(k) + \eta_k \left(\tilde{i}^k - \sum_{\nu=0}^{m} V_\nu(k) \varphi_\nu(\underline{x}^k) \right) \varphi_\mu(\underline{x}^k),$$

przy czym oznaczenie $\tilde{i}^k$ użyto do zasygnalizowania, że poprawne rozpoznania i^k zawarte w ciągu uczącym zostały *dostosowane* do zadania dychotomii:

$$\tilde{i}^k = \begin{cases} +1, & \text{gdy} \quad i^k = i_1, \\ -1, & \text{gdy} \quad i^k = i_2 \end{cases}.$$

Algorytm dla tej metody zostanie pominięty, jako że jest on (z dokładnością do kilku mało istotnych szczegółów) prostszym wariantem algorytmu opisanego w podrozdziale 7.2.

Przykład. Oba omówione warianty metody funkcji potencjalnych i metodę aproksymacji stochastycznej zastosowano do rozpoznawania skuteczności prze-

Tabela 7.1. Porównanie szybkości działania i dokładności kilku wybranych metod rozpoznawania obrazów

Metoda	Rozpoznawanie	
	czas, ms	dokładność, %
Funkcji potencjalnych 1	21,2	64,3
Funkcji potencjalnych 2	2,1	62,8
Aproksymacji stochastycznej	2,6	65,2
NN	20,7	87,4
NM	1,9	58,2
Funkcji liniowych	0,8	61,4
Bayesa	2,4	72,4
Parzena	76,5	64,8

ciwwirusowego działania wybranych tiosemikarbazonów [62]. Podano informacje o szybkości działania i o dokładności obydwu wymienionych metod, przytaczając także (dla porównania) wyniki uzyskane za pomocą innych metod([4]) - tabela 7.1.

7.4. Sieci neuronowe

Problematyka sieci neuronowych jest głównym przedmiotem innych książek (por. [50], [51], [52]), w związku z czym będzie tu przedstawiona w skrócie. Sieć neuronowa jest systemem([5]) przetwarzającym informacje w sposób równoległy. System ten składa się z dużej liczby elementów modelujących (w sposób bardzo uproszczony!) działanie rzeczywistych *neuronów*, formujących w mózgu (między innymi) struktury rozpoznające obrazy. Każdy z neuronów ma wiele wejść (*synaps*) i jedno wyjście (*akson*). O właściwościach poszczególnych neuronów decydują współczynniki $V_\nu^{(\lambda)}$ nazywane *wagami synaptycznymi*. Charakteryzują one każde wejście (o numerze ν) każdej komórki sieci (numerowanej przez λ). Dodatkowy współczynnik $V_0^{(\lambda)}$ nazywany jest *progiem* odpowiedniego neuronu. W celu opisu procesów w pojedynczym węźle sieci wprowadzamy pojęcie sumarycznego pobudzenia neuronu podczas pokazu k-tego elementu ciągu uczącego:

$$\tilde{e}_\lambda^k = \sum_{\nu=1}^{n} V_\nu^{(\lambda)(k-1)} x_\nu^k + V_0^{(\lambda)},$$

które wiążemy z jego sygnałem wyjściowym za pomocą funkcji nieliniowej φ:

$$\tilde{x}_\lambda^k = \varphi\left[\tilde{e}_\lambda^k\right].$$

Funkcja φ nie może być wybierana całkiem dowolnie, gdyż musi podlegać różniczkowaniu przy obliczaniu strategii uczenia. Wygodną postacią

([4]) Metody Bayesa i Parzena omówiono w rozdziale 8.

([5]) System ten może być wykonany w postaci specjalizowanego układu elektronicznego (na przykład Neurocomputer firmy ANZA), lecz częściej bywa realizowany w postaci programu symulacyjnego na dowolnym komputerze (na przykład pakiet NeuralWare).

funkcji φ jest krzywa logistyczna:

$$\tilde{x}_\lambda^k = \varphi\left[\tilde{e}_\lambda^k\right] = \frac{1}{1+\exp\left[-\sum_{\nu=1}^{n} V_\nu^{(\lambda)(k-1)} x_\lambda^k + V_0^{(\lambda)}\right]}.$$

Obok innych korzystnych właściwości krzywa ta ma także i tę zaletę, że jej pochodna ma bardzo prostą postać:

$$\frac{\partial \tilde{x}_\lambda^k}{\partial \tilde{e}_\lambda^k} = \tilde{x}_\lambda^k(1-\tilde{x}_\lambda^k).$$

Sieć neuronowa ma zwykle budowę warstwową. Wyróżnia się w niej warstwę wejściową, do której podawane są zewnętrzne sygnały (przy rozpoznawaniu obrazów są to wektory cech $\underline{x}$) oraz warstwę wyjściową, której sygnały mogą wskazywać na rozpoznanie i. Niech składowa x_ν wektora cech $\underline{x}$ będzie sygnałem wejściowym podawanym z zewnątrz do ν-tego wejścia każdego neuronu pierwszej warstwy, zaś $\tilde{x}_\lambda$ będzie sygnałem wyjściowym z λ-tego (w rozumieniu przyjętej numeracji) neuronu sieci. Proces uczenia sieci polega na prezentowaniu sieci w kolejnych krokach k wzorców tworzących zbiór uczący U i na dokonywaniu oceny wypracowywanych przez sieć sygnałów wyjściowych:

$$W = \{\tilde{x}_\lambda^k, \quad k = 1, 2, \ldots, N; \quad \lambda \in \mathcal{N}\}$$

porównywanych ze wzorcem, do którego sieć ma dążyć

$$Z = \{\hat{x}_\lambda^k, \quad k = 1, 2, \ldots, N; \quad \lambda \in \mathcal{N}\}.$$

Współczynniki $V_\nu^{(\lambda)}$ zmieniają się w trakcie prezentacji sygnałów $\tilde{x}_\nu^k$ o wielkość[6] $\Delta V_\nu^{(\lambda)(k)}$, przy czym reguła uczenia dana jest wzorem:

$$\Delta V_\nu^{(\lambda)(k)} = \zeta_1\left(\hat{x}_\lambda^k - \tilde{x}_\lambda^k\right)\frac{\mathrm{d}\varphi}{\mathrm{d}\tilde{e}_\lambda}^k x_\nu^k,$$

---

[6] Wielkość ta zależy od arbitralnie przyjmowanego współczynnika ζ_1.

co można zapisać w postaci:

$$\Delta V_{\nu}^{(\lambda)(k)} = \zeta_1 \delta_{\lambda}^{(k)} x_{\nu}^{k},$$

$$\delta_{\lambda}^{(k)} = (\hat{x}_{\lambda}^{k} - \tilde{x}_{\lambda}^{k}) \frac{\mathrm{d}\varphi}{\mathrm{d}\tilde{e}_{\lambda}^{k}}.$$

Po podstawieniu błąd $\delta_{\lambda}^{(k)}$ może być wyrażony wygodnym wzorem:

$$\delta_{\lambda}^{(k)} = (\hat{x}_{\lambda}^{k} - \tilde{x}_{\lambda}^{k})\, \tilde{x}_{\lambda}^{k} (1 - \tilde{x}_{\lambda}^{k}).$$

Podany algorytm uczenia można wyprowadzić z żądania minimalizacji średniokwadratowego błędu funkcjonowania sieci. Definiując kryterium stopnia niedopasowania na kroku k jako

$$Q_k = \frac{1}{2} \sum_{\lambda} (x_{\lambda}^{k} - x_{\lambda}^{k})^2$$

łączne kryterium niedopasowania $\tilde{x}$ do $\hat{x}$ wyrazić można następująco:

$$Q = \sum_{k=1}^{N} Q_k.$$

Możemy na początku poszukiwać minimum każdej składowej funkcji kryterialnej Q_k, a potem złożymy wyniki. Posłużymy się przy tym znaną techniką gradientową:

$$\Delta V_{\nu}^{(\lambda)(k)} = -\zeta_1 \frac{\partial Q_k}{\partial V_{\nu}^{(\lambda)}}.$$

Wartości pochodnych cząstkowych wyznaczamy jako

$$\frac{\partial Q_k}{\partial V_{\nu}^{(\lambda)}} = \frac{\partial Q_k}{\partial \tilde{x}_{\lambda}^{k}} \frac{\partial \tilde{x}_{\lambda}^{k}}{\partial V_{\nu}^{(\lambda)}}.$$

Ze wzoru definiującego funkcję Q_k wyznaczymy pochodną cząstkową

$$\frac{\partial Q_k}{\partial \tilde{x}_{\lambda}^{k}} = -(\hat{x}^{k} - \tilde{x}^{k}),$$

a ostatnią pochodną obliczymy jako

$$\frac{\partial Q_k}{\partial V_\nu^{(\lambda)}} = \frac{\partial Q_k}{\partial \tilde{e}_\lambda^k} \frac{\partial \tilde{e}_\lambda^k}{\partial V_\nu^{(\lambda)}} = \frac{\partial Q_k}{\partial \tilde{x}_\lambda^k} \frac{\partial \tilde{x}_\lambda^k}{\partial \tilde{e}_\lambda^k} \frac{\partial e_\lambda^k}{\partial V_\nu^{(\lambda)}} = \frac{\partial Q_k}{\partial \tilde{x}_\lambda^k} \frac{\mathrm{d}\varphi}{\mathrm{d}\tilde{e}_\lambda^k} \frac{\partial e_\lambda^k}{\partial V_\nu^{(\lambda)}}.$$

Przy okazji warto zauważyć, że strategia uczenia w czasie prezentacji wszystkich zestawów sygnałów treningowych x_ν^k, $k = 1, 2, \ldots, N$ może być określona jako prosta zasada sumowania poprawek $\Delta V_\nu^{(\lambda)(k)}$ dla wszystkich wartości k:

$$V_\nu^{(\lambda)} = V_\nu^{(\lambda)(0)} + \sum_{k=1}^{N} \Delta V_\nu^{(\lambda)(k)},$$

co jest naturalną konsekwencją wprowadzonej addytywnej postaci funkcji Q i wynikającej z niej oczywistej tożsamości:

$$\frac{\partial Q}{\partial V_\nu^k} = \sum_{k=1}^{N} \frac{\partial Q_k}{\partial V_\nu^{(\lambda)}}.$$

Można udowodnić, że proces uczenia jest zbieżny, jeśli tylko stawiane zadanie przekształcenia sygnałów x_ν w sygnały x_λ jest wykonalne w klasie zadań realizowalnych przez sieci neuropodobne. Co więcej, można udowodnić, że właściwe wartości współczynników $V_\nu^{(\lambda)}$ możliwe są do ustalenia po *skończonej* liczbie N pokazów (por. Dodatek 2). Opisany algorytm można jednak stosować jedynie w przypadku, kiedy $\tilde{x}_\lambda^k$ jest sygnałem z *wyjściowej* warstwy sieci – gdyż tylko wówczas znany jest sygnał $\hat{x}_\lambda^k$.

W przypadku rozważania sieci wielowarstwowych obecność warstw wewnętrznych (ukrytych) zmusza do modyfikacji algorytmu, ponieważ sygnały wejściowe większości neuronów są sygnałami wyjściowymi innych komórek, przeto w odpowiednich zapisach pojawią się sygnały $\tilde{x}_\lambda^k$ zamiast x_λ^k, na przykład we wzorze określającym pobudzenie komórki:

$$\tilde{e}_\lambda^k = \sum_{\nu=1}^{n} V_\nu^{(\lambda)(k-1)} \tilde{x}_\nu^k + V_0^{(\lambda)}$$

lub w stale aktualnym wzorze opisującym podstawową regułę uczenia komorek sieci (także wielowarstwowej):

$$\Delta V_{\nu}^{(\lambda)(k)} = \zeta_1 \delta_{\lambda}^{(k)} \tilde{x}_{\nu}^{k}.$$

Jak wykazano w pracach Rumelharta i współautorów [63], w sieciach tego typu współczynnik określający błąd popełniany przez rozważany λ-ty neuron $\delta_{\lambda}^{(k)}$ może być obliczony z *rekursywnych* formuł:

$$\delta_{\lambda}^{(k)} = (\hat{x}_{\lambda}^{k} - \tilde{x}_{\lambda}^{k}) \frac{d\varphi}{d\tilde{e}_{\lambda}^{k}},$$

jeśli λ-ty neuron należy do wyjściowej warstwy, albo ze wzoru:

$$\delta_{\lambda}^{(k)} = \frac{d\varphi}{d\tilde{e}_{\lambda}^{k}} \sum_{\mu} V_{\lambda}^{(\mu)(k)} \delta_{\mu}^{(k)},$$

gdy λ-ty neuron należy do warstwy ukrytej, ale połączony jest z neuronami o numerach μ warstw, dla których już ustalono wartości błędów $\delta_{\mu}^{(k)}$. Warto zwrócić uwagę, że są to zawsze neurony *dalsze* w sensie kierunku przepływu sygnału, zatem następuje tu proces rzutowania błędów wstecz – stąd nazwa *back propagation*, jaka została związana z tą metodą. Warto także odnotować, że w budowie sygnału $\delta_{\lambda}^{(k)}$ uczestniczą składowe $\delta_{\mu}^{(k)}$ tylko tych μ-tych neuronów, które są bezpośednio połączone z rozważanym λ-tym neuronem.

Algorytm dla metody sieci neuronowych pominięto, ponieważ zalety tego podejścia ujawniają się jedynie wtedy, gdy stosowana jest sprzętowa (w pełni równoległa) realizacja procesu uczenia i rozpoznawania, zaś symulacje komputerowe nie mają istotnych walorów, a odznaczają się bardzo dużą złożonością.

8. METODY PROBABILISTYCZNE

8.1. Postawienie zadania i podstawowe założenia

Omówimy teraz grupę metod opierających się na podejściu do definicji zadania rozpoznawania całkowicie odmiennym od przedstawionego, jakkolwiek – co zostanie wykazane – w wielu przypadkach prowadzącym do podobnych wyników. W szczególności różnica polega na sposobie traktowania zbioru uczącego U, który w omawianych metodach traci na znaczeniu.

Punktem wyjścia w rozważanych tu metodach rozpoznawania, zwanych dalej *probabilistycznymi*, są informacje o charakterze *statystycznym*. Źródło ich pochodzenia jest przy tym obojętne; w samych metodach określone prawdopodobieństwa traktuje się jako *dane*. W praktyce potrzebne rozkłady usiłuje się niekiedy estymować na podstawie ciągu uczącego U, ale trudności z tym związane (dalej wyczerpująco omówione) powodują, że przydatność praktyczna metod probabilistycznych jest generalnie mniejsza, niż można z pozoru oczekiwać(1).

Niech będą więc dane aprioryczne prawdopodobieństwa występowania obiektów należących do poszczególnych klas: $p^1, p^2, \ldots, p^L$, przy czym

$$\forall_{i \in I} \left[p^i = Prawd(d \in D^i) \right]. \tag{81}$$

---

(1) Trudności te nie występują w tych obszarach zastosowań rozpoznawania obrazów, w których tradycyjnie gromadzi się i opracowuje dane statystyczne, w wyniku czego potrzebne rozkłady prawdopodobieństwa można traktować rzeczywiście jako dane. Obszarami tymi są – przykładowo – badania ekonomiczne i medycyna.

Niech będą dane warunkowe rozkłady gęstości prawdopodobieństwa $P(\underline{x}/1), P(\underline{x}/2), \ldots, P(\underline{x}/L)$, rozumiane jako prawdopodobieństwa([2]) wystąpienia wektora $\underline{x}$ przy założeniu, że obiekt należy do klasy i

$$\forall_{\underline{\mu} \in X} \left[\forall_{\nu \in I} [P(\underline{\mu}/\nu) = Prawd(i = \nu \Rightarrow \underline{x} = \mu)] \right]. \tag{82}$$

Wprowadźmy pojęcie błędu([3]) rozpoznawania $b_{\mu\eta}$, którego indeksami są: rzeczywista przynależność obiektu μ oraz przynależność η rozpoznana przez algorytm $\hat{A}$ (patrz (7)). Załóżmy, że dopuszczalne (z punktu widzenia wymagań użytkownika systemu rozpoznającego) prawdopodobieństwo błędu $b_{\mu\eta}$ oznaczać będziemy przez $e_{\mu\eta}$. Przy metodach rozpoznawania etapowego $e_{\mu\eta}$ (zadawane jako parametr procedury rozpoznającej([4])) jest podstawą do podjęcia decyzji – czy należy określać wartości kolejnych cech x_j, czy też można już próbować ustalić przynależność rozpoznawanego obiektu. Niech ponadto dla każdego $b_{\mu\eta}$ dana będzie liczba([5]) $q_{\mu\eta}$ określająca stratę, związaną z tym błędem:

$$\forall_{\mu \in I} [\forall_{\eta \in I} [q_{\mu\eta} \in \mathcal{R}]]. \tag{83}$$

Odnośnie do strat $q_{\mu\eta}$ można zakładać, że

$$q_{\mu\mu} = \min_{\eta \in I} q_{\mu\eta} \tag{84}$$

oraz na ogół

$$q_{\mu\eta} \neq q_{\eta\mu}. \tag{85}$$

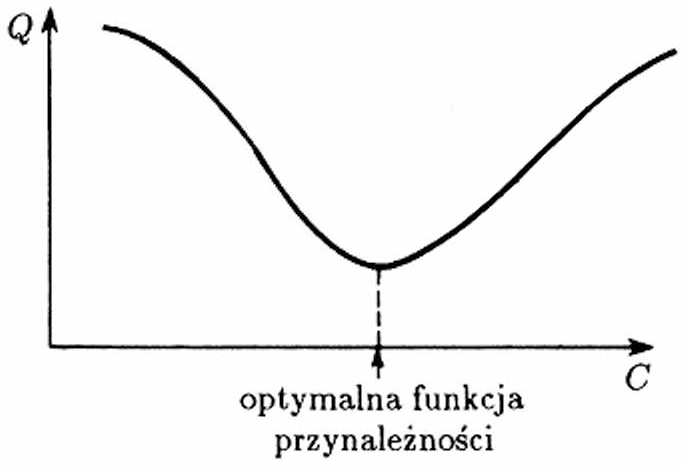
([2]) Można je wyrazić formułą alternatywną w stosunku do (82)

$$\forall_{\underline{x} \in X} [\forall_{i \in I} [P(\underline{x}/i) = Prawd(d \in D^i \Rightarrow B(d) = \underline{x})]].$$

([3]) Dla jednolitości notacji przyjmuje się także istnienie „błędów" typu $b_{\mu\mu}$ (czyli poprawnych ropoznań).

([4]) Wartości $e_{\mu\eta}$ narzucane są przez użytkownika metody na ogół w sposób całkowicie arbitralny.

([5]) Warto zauważyć, że wprowadzenie takiej liczby prowadzi do sformułowania zadania optymalnego rozpoznawania, co stanowi nowość w porównaniu z dotychczas omawianymi metodami.



Rys. 8.1. Wartość funkcji kryterialnej Q zależy od wybranej postaci funkcji przynależności C. Zwykle zależność ta ma minimum, którego osiągnięcie można utożsamiać z wyborem optymalnej techniki rozpoznawania

Na podstawie przyjętych założeń, charakteryzujących zadanie klasyfikacji A (por. (14)), można budować ocenę jakości $Q(A, \hat{A})$ algorytmu rozpoznającego $\hat{A}$, a następnie można poszukiwać takiej formy funkcji przynależności $C^i(\underline{x})$, aby minimalizować $Q(A, \hat{A})$ (rys. 8.1).

Budowę oceny $Q(A, \hat{A})$ przeprowadzimy etapowo. Najpierw zdefiniujemy oczekiwaną wartość oceny (straty) dla pewnego ustalonego obiektu $\underline{x}$ przy przyjęciu pewnej, ustalonej decyzji i. Oznaczając tę stratę przez $Q^i(x)$, możemy zapisać

$$Q^i(\underline{x}) = \sum_{\nu=1}^{L} q_{\nu i} P(\nu/\underline{x}). \tag{86}$$

Prawdopodobieństwo $p(\nu/\underline{x})$ określające, że mamy do czynienia z obiektem klasy $\nu \in I$, jeśli zaobserwowano wektor cech $\underline{x} \in X$, obliczyć można na podstawie przytoczonych danych, opierając się na wzorze Bayesa:

$$p(\nu/\underline{x}) = \frac{P^\nu p(\underline{x}/\nu)}{\sum_{i=1}^{L} p^i P(\underline{x}/i)} = \frac{p^\nu p(\underline{x}/\nu)}{p(\underline{x})}. \tag{87}$$

Na podstawie znajomości strat warunkowych $Q^i(\underline{x})$ oraz apriorycznych prawdopodobieństw klas p^i obliczyć można uogólnioną stratę $Q(\underline{x})$ oce-

kiwaną w przypadku pojawienia się obiektu opisywanego zbiorem cech $\underline{x}$:

$$Q(\underline{x}) = \sum_{i=1}^{L} p^i Q^i(\underline{x}) = \sum_{i=1}^{L} p_i \sum_{\nu=1}^{L} q_{\varphi i} p(\nu/\underline{x}). \tag{88}$$

Ogólną ocenę $Q(A, \hat{A})$ uzyskać można, uniezależniając ocenę (88) od konkretnego obiektu $\underline{x}$ poprzez całkowanie po całej przestrzeni X:

$$Q(A, A) = \int\int \cdots \int_{n} Q(\underline{x}) p(\underline{x}) \, \mathrm{d}\underline{x}. \tag{89}$$

Algorytm rozpoznawania $\hat{A}$ jest wybierany w ten sposób, aby minimalizował funkcję kryterialną $Q(A, \hat{A})$. Na ogół dokonuje się tego poprzez minimalizację funkcji podcałkowej. Zadanie polega więc na wyznaczeniu algorytmu rozpoznawania zapewniającego minimum funkcji (88). Zadanie to można stosunkowo łatwo rozwiązać w przypadkach szczególnych, natomiast ogólne rozwiązanie jest trudne do konstruktywnego sformułowania.

8.2. Metoda rozpoznawania w przestrzeni jednowymiarowej

Rozważymy instruktywny przypadek szczególny. Niech

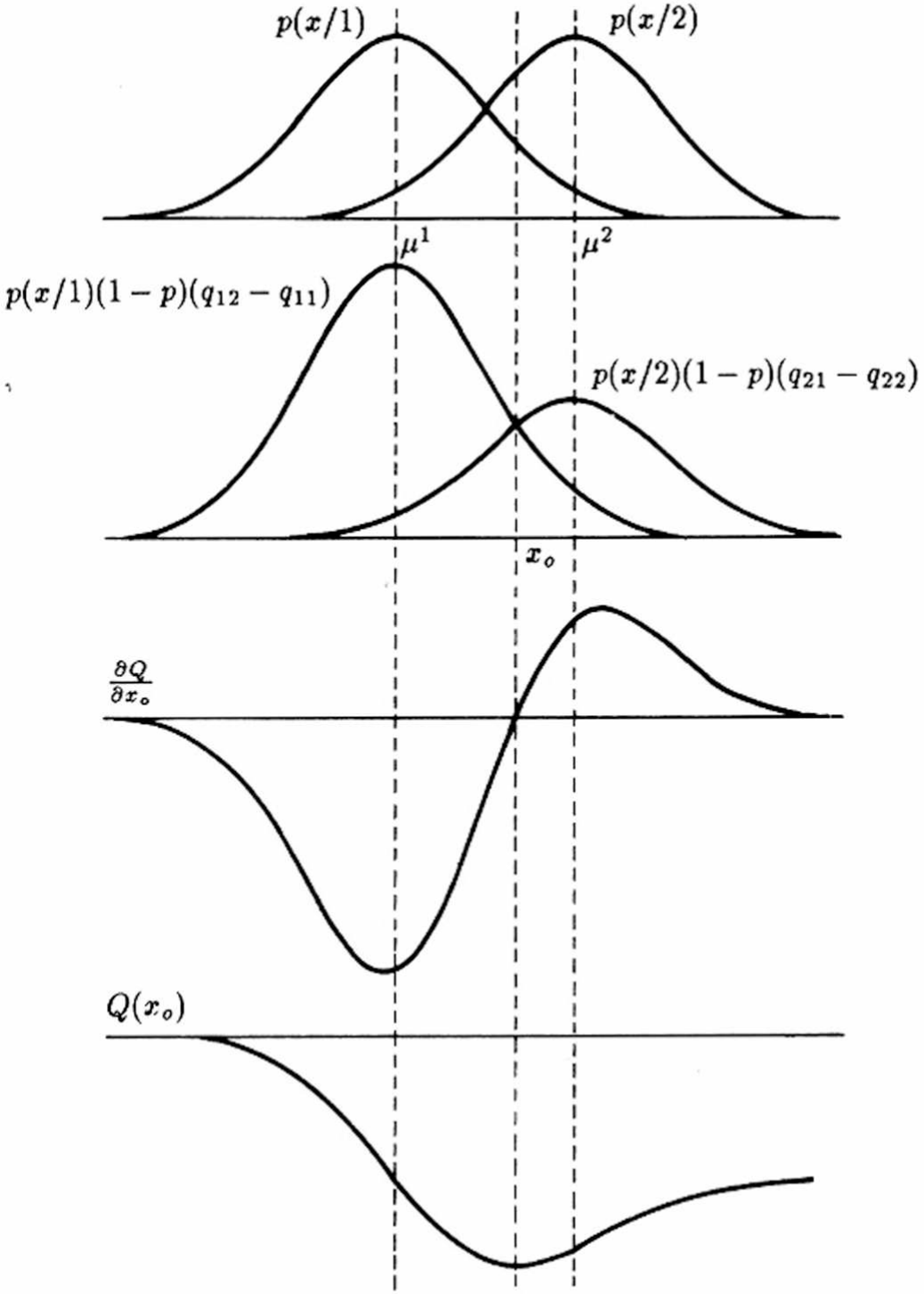
$$L = 2;\ n = 1;\ p^2 = p;\ p^1 = 1 - p;\ 0 < p < 1; \tag{90}$$

zaś rozkłady

$$P(x/1) = \frac{1}{\sqrt{2\pi\eta}} \exp\left[-\frac{(x - \mu^{(1)})^2}{2\eta^2}\right], \tag{91}$$

$$P(x/2) = \frac{1}{\sqrt{2\pi\eta}} \exp\left[-\frac{(x - \mu^{(2)})^2}{2\eta^2}\right], \tag{92}$$

niech będą jednowymiarowymi rozkładami Gaussa o zgodnych wariancjach; zakładamy również znajomość parametrów p, $\mu^{(1)}$, $\mu^{(2)}$, q_{11}, q_{12}, q_{21}, q_{22} (rys 8.2).



Rys. 8.2. Ilustracja zadania rozpoznawania w przestrzeni jednowymiarowej

Niech ponadto poszukiwana reguła decyzyjna ma postać:

$$\hat{A}(x) = \begin{cases} 1, & \text{gdy} \quad x < x_0, \\ 2, & \text{gdy} \quad x \geq x_0 \end{cases} \tag{93}$$

(przy oczywistym założeniu, że $\mu^{(2)} > \mu^{(1)}$). Wówczas

$$\begin{aligned} Q(A, \hat{A}) = (1-p)\left[q_{11} \int_{-\infty}^{x_0} P(x/1)\,dx + q_{12} \int_{x_0}^{\infty} P(x/1)\,dx\right] + \\ + p\left[q_{22} \int_{x_0}^{\infty} P(x/2)\,dx + q_{21} \int_{-\infty}^{x_0} p(x/2)\,dx\right]. \end{aligned} \tag{94}$$

Z zapisu wzoru (94) widać, że

$$Q(A, \hat{A}) = Q(x_0) \tag{95}$$

i możliwa jest próba bezpośredniej optymalizacji

$$\begin{aligned} \frac{\partial Q(x_0)}{\partial x_0} = (1-p)[q_{11}P(x_0/1) - q_{12}P(x_{12}/1)] + \\ + P[q_{21}p(x_0/2) - q_{22}P(x_0/2)] = 0. \end{aligned} \tag{96}$$

Rozwiązując równanie (96) dochodzimy do wzoru([6]) (przy oczywistym założeniu, że $q_{22} < q_{21}$):

$$\frac{P(x_0/2)}{P(x_0/1)} = \frac{(1-p)(q_{12} - q_{11})}{p(q_{21} - q_{22})}. \tag{97}$$

Wykorzystując (91) i (92), można dalej obliczyć, że

$$x_0 = \frac{\mu^{(1)} + \mu^{(2)}}{2} - \frac{\eta^2}{\mu^1 - \mu^2} \ln \frac{(1-p)q_{12}}{pq_{21}}. \tag{98}$$

([6]) Zauważmy, że w rozważanym wzorze pojawia się po raz pierwszy iloraz prawdopodobieństw, który odegra ważną rolę przy omawianym dalej rozpoznawaniu etapowym.

Postać ostatecznej formuły (98) jest w gruncie rzeczy mało istotna, gdyż warunki (90), (91) i (92) wyznaczają mało realistyczne z praktycznego punktu widzenia zadania rozpoznawania. (Szczególnie założenie $n = 1$, dzięki któremu udało się odpowiednie rachunki przeprowadzić efektywnie do końca i zinterpretować je graficznie na rysunku 8.2, jest założeniem dyskwalifikującym praktyczną przydatność wyniku (98)). Ważne jest natomiast stwierdzenie faktu, że formułę (98) dało się bezpośrednio wyprowadzić z warunku:

$$Q(A, \hat{A}) = \min, \tag{99}$$

co prowadzi do wniosku, że takie postawienie zadania ma sens, a ewentualne trudności są jedynie rachunkowej natury. Warto jednak zwrócić uwagę na kształt formuły (97). Wynika z niej, że stosunek prawdopodobieństw obliczanych w punkcie odpowiadającym granicy obszarów, w których podejmowane są odmienne decyzje, jest determinowany przez pewną stałą zależną od parametrów zadania. Jest to ważny fakt, mający znaczenie ogólne (to znaczy nie wynikający jedynie z założeń $n = 1$; $L = 2$), do którego będziemy dalej wielokrotnie wracali.

Przykład. Zadanie rozpoznawania w przestrzeni jednowymiarowej bywa niekiedy możliwe do praktycznego zastosowania po odpowiedniej redukcji przestrzeni cech. Przykładem takiej sytuacji mogą być prace [64] zmierzające do rozpoznawania stopnia zjadliwości określonych szczepów bakteryjnych. W zadaniu tym dzięki zastosowaniu metody składowych kanonicznych udało się wielowymiarowy wektor obserwacji prowadzonych na zwierzętach i hodowlach tkankowych zamienić najpierw na układ zdekorelowanych składowych kanonicznych, a następnie – wybierając tę spośród składowych, która niosła ponad 80% użytecznej informacji – udało się ograniczyć problem podejmowania decyzji do analizy wartości jednej tylko składowej.

Algorytm rozpoznawania w przestrzeni jednowymiarowej jest bardzo prosty i wynika w sposób natychmiastowy z przytoczonych wzorów.

```
begin
    GetParameters;    {wczytanie parametrów: p, μ(1),
                       μ(2), η, q11, q12, q21, q22}
    threshold := CountThreshold;    {obliczenie ze wzoru (98)}
    if obj > threshold   {obj jest w tym przypadku prostą zmienną}
      then rec := 2
      else rec := 1;
end
```

8.3. Rozpoznawanie w przestrzeni wielowymiarowej

Wychodząc w rozważaniach poza prosty przypadek ($n = 1$, $L = 2$), musimy oprzeć się na następującym rozważaniu upraszczającym.

Całkę (89) możemy zminimalizować, dobierając dla każdego rozpoznawanego obiektu $\underline{x}$ numer rozpoznawanej klasy i tak, aby funkcja podcałkowa (88) osiągała minimum. Z kolei minimalizacja funkcji $Q(\underline{x})$ może być osiągnięta wtedy, gdy dla każdego obiektu $\underline{x}$ wybierana będzie taka decyzja i, dla której ocena straty (86) będzie minimalna. Tak więc należy wybierać ten numer klasy i, który odpowiada minimalnej wartości $Q^i(\underline{x})$. Funkcja $Q^i(\underline{x})$ pełni więc rolę *odwrotną* do funkcji przynależności $C^i(\underline{x})$. Jeżeli teraz podstawimy (87) do (86), to otrzymamy przydatny w dalszej analizie wzór:

$$Q^i(\underline{x}) = \frac{1}{\sum_{\mu=1}^{L} p^\mu P(\underline{x}/\mu)} \sum_{\nu=1}^{L} p^\nu P(\underline{x}/\nu) q_{\nu i}. \tag{100}$$

Zauważmy, że $\sum p^\mu p(\underline{x}/\mu)$ nie zależy od numeru klasy i, w związku z czym jest on jednakowy dla wszystkich $Q^i(\underline{x})$ i *nie wpływa na decyzję*. Usuwając wskazany czynnik, otrzymamy uproszczony wzór:

$$\hat{Q}^i(\underline{x}) = \sum_{\nu=1}^{L} p^\nu P(\underline{x}/\nu) q_{\nu i}. \tag{101}$$

W celu efektywnego prowadzenia dalszych obliczeń musimy założyć coś o charakterze zależności $q_{\mu\eta}$ (por. (83), (84), (85)). Niech

$$q_{\mu\eta} = 1 - \delta_{\mu\eta}, \tag{102}$$

gdzie $\delta_{\mu\eta}$ jest wprowadzoną wcześniej funkcją zgodności Kroneckera (por. wzór (32)). Warunek (102) jest często nazywany warunkiem symetrycznej funkcji strat, gdyż

$$q_{\mu\eta} = \begin{cases} 1 & \text{dla} \quad \mu \neq \eta, \\ 0 & \text{dla} \quad \mu = \eta. \end{cases} \tag{103}$$

Wykorzystując własność (102) lub (103) we wzorze (101), stwierdzamy, że

$$\hat{Q}^i(\underline{x}) = \sum_{\nu=1}^{L} p^\nu P(\underline{x}/\nu). \tag{104}$$

Ponownie znajdujemy we wzorze (104) składnik $\sum p^\nu P(\underline{x}/nu)$, o którym wcześniej stwierdzono, że nie zależy od i. Przeto można zauważyć, że $\hat{Q}^i(\underline{x})$ będzie tym mniejsze, im większy będzie składnik $p^i p(\underline{x}/i)$. Składnik ten można więc utożsamić z funkcją przynależności

$$C^i(\underline{x}) = p^i P(\underline{x}/i), \tag{105}$$

której *maksimum* wskazywać będzie właściwą klasę i. Prostota wzoru (105) decyduje o jego popularności; zauważając dodatkowo, że dowolna, monotonicznie rosnąca funkcja argumentu $C^i(x)$ może być również używana w charakterze funkcji przynależności, możemy wzór (105) poddać obustronnemu logarytmowaniu, otrzymując formułę:

$$C^i(x) = \ln P(x/i) + \ln p^i \tag{106}$$

bardzo przydatną w analizie przypadków szczegółowych.

Przykład. W wielokrotnie cytowanej rozprawie doktorskiej L. Kota użyto wzoru (106) do rozpoznawania samogłosek. Uzyskano następujące odsetki poprawnych rozpoznań – tabela 8.1 Ponieważ w tym samym zadaniu metody minimalnoodległościowe dawały zawsze 100% poprawnych rozpoznań – nie jest to wynik świadczący o zaletach metod probabilistycznych!

Tabela 8.1. Poprawność rozpoznawania samogłosek za pomocą metod probabilistycznych

Samogłoska	i	y	e	e	o	u
Procent rozpoznań	100	97,0	82,3	95,8	95,5	81,1

Przedstawmy teraz rozważaną metodę w postaci ogólnego algorytmu. Będą w nim określone następujące obiekty:

Prob[1 . . **numclass**] – tablica prawdopodobieństw klas (p^i),

density(i, obj) – funkcja określająca warunkowe gęstości ($p(x/i)$).

```
begin
    for i := 1 to numclass do
    fun[i] := log ( density(i, obj)) + log(Prob[i])
    rec := pointmax(fun);
end
```

8.4. Określenie wymaganych rozkładów prawdopodobieństwa

Przystępując do analizy wybranych szczegółowych zagadnień związanych z wykorzystaniem probabilistycznych metod rozpoznawania musimy wrócić do definicji (82) i stwierdzić, że w praktyce znajomość rozkładów $P(\underline{x}/i)$ dla wszystkich klas i jest problematyczna. W dodatku zauważmy, że wzór (106) (i poprzednie) wymaga znajomości postaci $P(\underline{x}/i)$ nadającej się do efektywnego obliczania przy dowolnym $\underline{x}$, czyli w rezultacie najchętniej – w postaci analitycznej (por. postulowaną funkcję density(i, obj)). Jest to bardzo poważny problem na gruncie statystyki: jak uzyskać rozkład *wielowymiarowy*([7]) zmiennej losowej $\underline{x}$ jedynie na podstawie obserwacji próbek $\langle x^k, i^k \rangle \in U$. W rozważanym zadaniu rozpoznawania problem jest jeszcze trudniejszy: potrzeba takich rozkładów (warunkowych!) nie jeden, lecz L.

Możliwe są tu trzy podejścia. Pierwsze zakłada, że rozkłady $P(\underline{x}/i)$ są znane i z góry szczegółowo zadane (na przykład: na podstawie wcześniejszych badań statystycznych, na których możemy się oprzeć). Wówczas naturalnie problem nie istnieje i wzór (106) jest efektywny.

Podejście drugie bazuje na hipotezie o określonym (zadanym z góry) charakterze rozkładu. Przykładowo (co dalej będzie szerzej dyskutowane) często można założyć, że interesujący rozkład jest rozkładem Gaussa lub Bernoulliego. Zadanie znalezienia rozkładu $P(\underline{x}/i)$ sprowadza się wówczas do znalezienia (estymacji) parametrów tego rozkładu. Tego typu problemy są szczegółowo omawiane na gruncie statystyki i mają swoje rozwiązania

---

([7]) Warto zauważyć, że w odróżnieniu od wcześniej dyskutowanego zagadnienia rozkładów $P(x/i)$; $i = 1, \ldots, L$, zagadnienie znalezienia wartości p^i, $i = 1, \ldots L$, jest w praktyce zupełnie proste. Wystarczy tu zwykle estymacja na podstawie frekwencji obiektów klasy i w ciągu uczącym U. Można przyjąć na przykład $p^i = N^i/N$ lub inną, równie prostą regułę.

łatwo dostępne w monografiach poświęconych estymacji parametrów statystycznych. Co więcej, dostępne są metody statystycznej *weryfikacji* hipotezy o określonym charakterze interesujących rozkładów, czyli zakładając, że $P(\underline{x}/i)$ jest rozkładem określonego rodzaju (na przykład normalnym), możemy to założenie *sprawdzić*. W praktyce weryfikacja charakteru rozkładu $P(\underline{x}/i)$ jest jedak uciążliwa i bywa zwykle pomijana, co stanowi błąd w świetle przytoczonych uwag.

Ostatnie – trzecie podejście – polega na próbie bezpośredniego (najczęściej iteracyjnego) budowania rozkładów $P(\underline{x}/i)$ bez wykorzystywania jakichkolwiek założeń dotyczących charakteru tego rozkładu. Podejście to także zostanie dalej przedstawione.

Bazując na drugim z wymienionych wyżej podejść, możemy dokonać analizy kilku praktycznie interesujących przypadków szczególnych.

8.5. Przypadek niezależnych składowych wektora cech

Załóżmy, że poszczególne składowe wektora $\underline{x}$ są zmiennymi losowymi binarnymi

$$\forall_{\nu\in[1,n]}[x_\nu \in \{0,1\}] \tag{107}$$

oraz statystycznie niezależnymi([8])

$$\begin{aligned}\forall_{\nu,\mu\in[1,n]}[&Prawd(x_\nu = \eta^1 \wedge x_\eta = \eta^2) = \\ &= Prawd(x_\nu = \eta^1) \cdot Prawd(x_\mu = \eta^2)].\end{aligned} \tag{108}$$

Wówczas oznaczając (por. wzór (82))

$$\forall_{\mu\in I}\left[\forall_{\nu\in[1,n]}[Prawd(\mu = i \Rightarrow x_\nu = 1) \equiv \eta^i]\right], \tag{109}$$

mamy oczywiście

$$Prawd(\mu = i \Rightarrow x_\nu = 0) = 1 - \eta^i_\nu. \tag{110}$$

([8]) Hipoteza niezależności składowych x_ν i x_μ jest bardzo silnym założeniem i powinna być bezwarunkowo weryfikowana statystycznie przed użyciem opisanej dalej metody. Do weryfikacji można użyć testu Chi-kwadrat lub testu Pearsona.

Biorąc to pod uwagę oraz wykorzystując fakt, że $x_\nu \in \{0, 1\}$, otrzymujemy:

$$P(\underline{x}/i) = (\eta_\nu^i)^{x_\nu}(1 - \eta_\nu^i)^{1-x_\nu}. \tag{111}$$

Wykorzystując postulowaną *niezależność* współrzędnych x_ν, mamy

$$P(\underline{x}/i) = \prod_{\nu=1}^{n} p(x_\nu/i) \tag{112}$$

i podstawiając (111) do (112), otrzymujemy

$$p(\underline{x}/i) = \prod_{\nu=1}^{n} \left[(\eta_\nu^i)^{x_\nu}(1 - \eta_\nu^i)^{(1-x^\nu)}\right]. \tag{113}$$

Zatem funkcja przynależności (106) ma w tym przypadku postać:

$$\tilde{C}^i(\underline{x}) = \sum_{\nu=1}^{n} \left[x \ln \eta_\nu^i + (1 - x_\nu)\ln(1 - \eta_\nu^i)\right] + \ln p^i. \tag{114}$$

Wzór (114) można przekształcić do wygodniejszej postaci:

$$\tilde{C}^i(\underline{x}) = \sum_{\nu=1}^{n} \left[\ln \eta_\nu^i - \ln(1 - \eta_\nu^i)\right] x_\nu + \sum_{\nu=1}^{n} (1 - \eta_\nu^i) + \ln p^i, \tag{115}$$

wskazującej, że funkcja przynależności jest w tym przypadku funkcją *liniową* (por. wzór (58)). W procesie uczenia opisanym w rozdziale 6 (por. wzory (66), (67), (68)) nie znane wartości η_ν^i oraz p^i nie występowały w sposób jawny i dlatego były wyznaczane niejako mimochodem, tu natomiast mogą być obliczone na podstawie metod statystyki, zgodnie z zasadą optymalizacji funkcji przynależności. Wniosek z przytoczonych rozważań jest dość zaskakujący. Jeśli spełnione są statystyczne założenia omawianego tu wariantu metody, to zadanie jest najwyraźniej liniowo separowalne, a wówczas można stosować albo metodę uczenia funkcji liniowych, albo obliczenia probabilistyczne – a efekt jest podobny.

Przedstawimy algorytm dla rozważanej tu wersji metody. Wprowadzimy następujące obiekty:

pr[1 . . numclass][1 . . dim] – prawdopodobieństwo, że dla ustalonej klasy określona cecha jest jedynką (η_ν^i)

```
begin
    for i := 1 to numclass do
    begin
      fun[i] := log(Prob[i]);
      for n := 1 to dim do
        fun[i] := fun[i] + (log(pr[i,n]) − log (1.0−pr[i,n]))*obj[n]
              + log(1.0 − pr[i,n]);
    end
    rec := pointmax(fun);
end
```

8.6. Przypadek wielowymiarowego rozkładu normalnego

Rozważmy z kolei inny przypadek. Niech rozkłady $p(\underline{x}/i)$ będą wielowymiarowymi rozkładami Gaussa([9])

$$P(\underline{x}/i) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{T}^i|}} \exp\left[-\frac{1}{2}(\underline{x} - \underline{M}^i)^T (\mathbf{T}^i)^{-1}(\underline{x} - \underline{M}^i)\right]. \quad (116)$$

Postać $P(\underline{x}/i)$ dla $L = 2$ przedstawiono przykładowo na rysunku 8.3. Przez M^i oznaczono wektor wartości oczekiwanych dla i-tej klasy, świadomie nawiązując do wprowadzonego wcześniej pojęcia wzorca (por. wzory (42) i (43)); zakładamy, że wartości elementów tego wektora będą obliczane jako średnie z elementów ciągu uczącego, należących do i-tej klasy,

([9]) W zastosowaniach praktycznych należy zawsze badać, czy hipoteza o normalności rozkładu jest spełniona z zadowalającym poziomem wiarygodności. Można do tego użyć uogólnionego na przypadek wielowymiarowy testu λ (Kołmogorowa), ale trzeba przyznać, że w ogólnym przypadku kontrola normalności rozkładu wielowymiarowego jest zadaniem bardzo trudnym. Dlatego dość często *próbujemy* zastosować podejście opisane w tym podrozdziale bez weryfikacji charakteru rozkładu – uzyskując nieraz dobre skutki w postaci poprawnego rozpoznawania nawet w przypadku, kiedy rozkład nie jest rozkładem normalnym, lecz dowolnym innym (byle unimodalnym o skończonej wariancji).

$$m_j^i = \frac{1}{N^i} \sum_{k=1}^{N^i} x_j^{i,k}. \tag{117}$$

Natomiast $\mathbf{T}^i$ jest macierzą kowariancji([10]) składowych wektora $\underline{x}$ dla i-tej klasy. Będą one wyznaczane ze wzoru:

$$\mathbf{T}_{\mu,nu}^i = \frac{1}{N^i - 1} \sum_{k=1}^{N^i} (x_\mu^{i,k} - m_\mu^i)(x_\eta^{i,k} - m_\eta^{i,k}). \tag{118}$$

Symbol $|\mathbf{T}^i|$ oznacza wyznacznik macierzy $\mathbf{T}^i$, $(\mathbf{T}^i)^{-1}$ jest macierzą odwrotną, a $(\underline{x} - \underline{M}^i)^T$ - wektorem transponowanym.

Wstawiając wzór (116) do (106), otrzymuje się bez trudu funkcję przynależności

$$\tilde{C}^i(\underline{x}) = -\frac{1}{2}(\underline{x} - \underline{M}^i)^T (\mathbf{T}^i)^{-1} (\underline{x} - \underline{M}^i) - \frac{1}{2} \ln \left[(2\pi)^n |\mathbf{T}^i|\right] + \ln p^i. \tag{119}$$

Łatwo pokazać, że funkcja (119) jest funkcją kwadratową ze względu na $\underline{x}$, przekształcając ją do postaci:

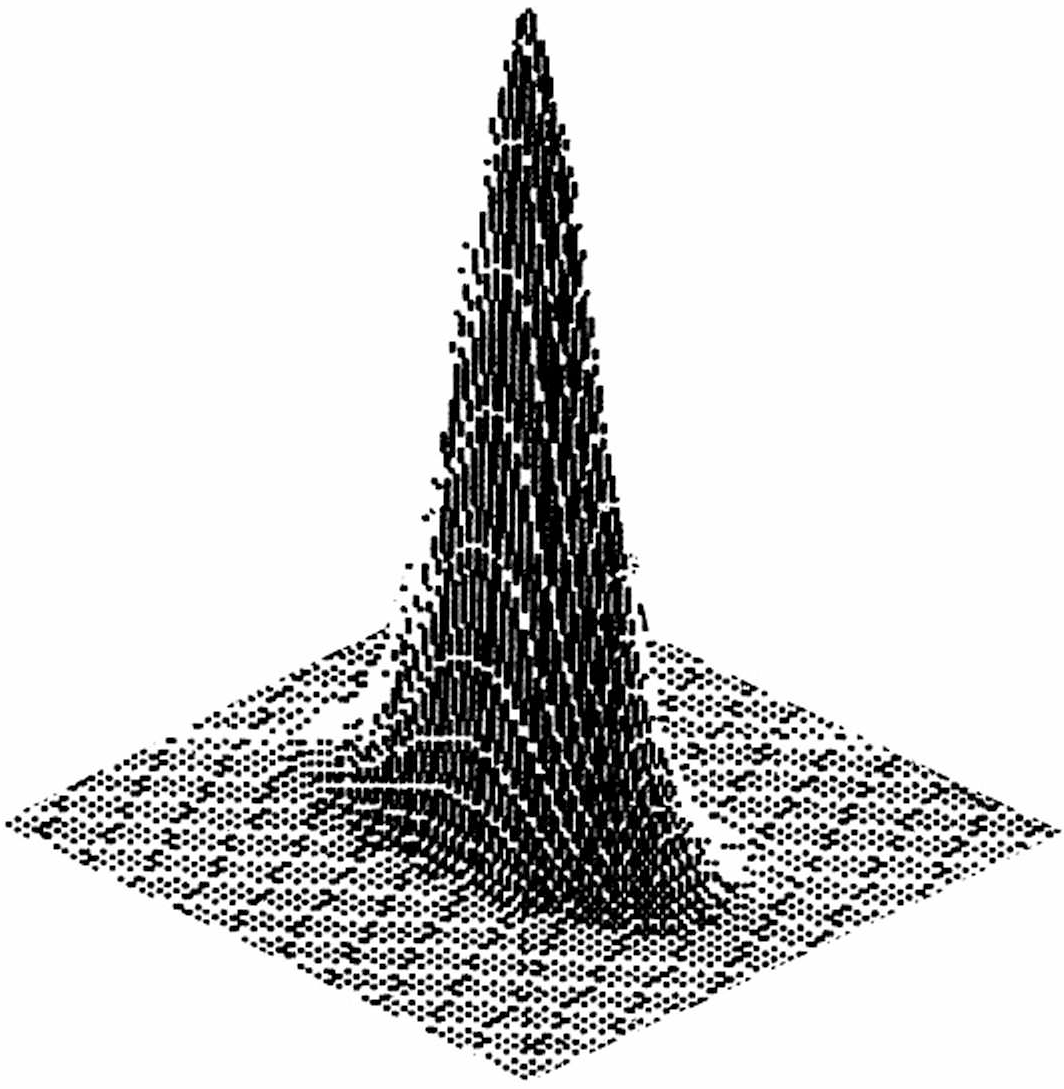
$$\begin{aligned} \tilde{C}^i(x) = &-\frac{1}{2}\underline{x}^T (\mathbf{T}^i)^{-1} \underline{x} + \underline{x}^T (\mathbf{T}^i)^{-1} \underline{M}^i - \frac{1}{2}(\underline{M}^i)^T (\mathbf{T}^i)^{-1} \underline{M}^i + \\ &+ \frac{1}{2} \ln \left[(2\pi)^n |\mathbf{T}^i|\right] + \ln p^i. \end{aligned} \tag{120}$$

Jeśli jednak założy się (co zbyt często w praktyce przyjmuje się nieco bezkrytycznie!), że macierze kowariancji nie różnią się w sposób istotny([11]), to znaczy zakładając

---

([10]) Dla poprawnego wyznaczenia elementów macierzy kowariancji konieczne jest dysponowanie dużą populacją (dużą liczbą obserwacji tworzących ciąg uczący N i każdy z podciągów N^i).

([11]) Badanie zgodności wariancji przeprowadzić można za pomocą odpowiedniego testu statystycznego, będącego uogólnieniem na przypadek wielowymiarowy statystyki **F** Fishera.



Rys. 8.3. Rozkład Gaussa dla $L = 2$

$$\mathbf{T}^1 = \mathbf{T}^2 = \cdots = \mathbf{T}^L = \mathbf{T}, \tag{121}$$

można wzór (120) znacznie uprościć, usuwając z niego wszystkie te elementy, które nie zależą od numeru klasy i, a zatem składniki (z punktu widzenia rozpoznawania) nie różnicujące. Otrzymuje się wówczas funkcję liniową

$$\hat{C}^i(x) = \underline{x}^T \mathbf{T}^{-1} \underline{M}^i - \frac{1}{2}(\underline{M}^i)^T \mathbf{T}^{-1} \underline{M}^i + \ln p^i, \tag{122}$$

wspomnianą już wcześniej wielokrotnie. Jak widać, w przypadku stosowania podejścia probabilistycznego i założenia normalnej postaci rozkładów $P(\underline{x}/i)$ funkcja przynależności przyjmuje formę liniową jedynie w przypadku jednakowych macierzy kowariancji.

Algorytm opisywanej metody jest strukturalnie bardzo prosty, ale jego pełny zapis jest dość długi ze względu na pracochłonne operowanie w komputerze formułami obliczeniowymi odwołującymi się do rachunku macierzowego. Z tego względu przedstawimy go w formie uproszczonej, wprowadzając dwie procedury (obliczające parametry statystyczne na podstawie ciągu uczącego) oraz trzy funkcje:

CountMean(mean) - obliczanie wektorów wartości średnich ($\underline{M}^i$),

CountVariance(variance) – obliczanie macierzy kowariancji wszystkich klas ($\mathbf{T}^i$),

EqualVariance(variance): Boolean – ocena wiarygodności hipotezy o identyczności macierzy kowariancji,

flin(i,obj): real – obliczenie funkcji przynależności według funkcji liniowej (wzór 122),

fsqr(i,obj): real – obliczanie funkcji przynależności według funkcji kwadratowej (wzór (120)).

Nowe tablice, używane w rozważanym algorytmie, a dotychczas nie deklarowane, są następujące:

mean[1 . . numclass][1 . . dim] – wektory średnie dla wszystkich klas,

variance[1 . . numclass][1 . . dim][1 . . dim] – macierze kowariancji wszystkich klas.

```
begin
      CountMean(mean); CountVariance(variance);
      for i := 1 to numclass do
        if EqualVaraiance(variance)
          then fun[i] := flin(i,obj)
          else fun[i] := fsqr(i,obj);
      rec := pointmax(fun);
end
```

8.7. Metody oparte na empirycznym budowaniu rozkładu

Przystąpimy do dyskusji trzeciego z wymienionych uprzednio podejść, opierając się na założeniu braku jakiejkolwiek (nawet strukturalnej) informacji o charakterze rozkładu $P(\underline{x}/i)$ i bazowaniu wyłącznie na ciągu uczącym U. Istnieje przy tym kilka rekomendowanych w literaturze metod, z których wybrano do omówienia nieparametryczną metodę Parzena oszacowania gęstości prawdopodobieństwa. Metoda ta bazuje na obliczaniu $P(\underline{x}/i)$ na podstawie wzoru:

$$P(\underline{x}/i) = \frac{1}{N^i} \sum_{k=1}^{N^i} \frac{1}{(\eta_k)^n} G\left(\frac{\underline{x} - \underline{x}^{i,k}}{\eta_k}\right), \tag{123}$$

gdzie η_k podobnie jak w metodzie funkcji potencjalnych oznacza uzbieżniający ciąg liczbowy, spełniający warunek:

$$[\eta > 0] \wedge \left[\lim_{k\to\infty} \eta_k = 0\right] \wedge \left[\lim_{k\to\infty} k(\eta_k)^n = \infty\right], \tag{124}$$

zaś jądro oszacowania G jest funkcją argumentu wektorowego $\underline{\mu}$, spełniającą warunki:

$$\begin{aligned} &\left[\int\int \cdots \int_n G(\underline{\mu})\, \mathrm{d}\underline{\mu} = 1\right] \wedge \left[\int\int \cdots \int_n |G(\underline{\mu})|\, \mathrm{d}\underline{\mu} < \infty\right] \wedge \\ &\wedge \left[\int\int \cdots \int_n G^2(\underline{\mu})\, \mathrm{d}\underline{\mu} < \infty\right] \wedge \left[\lim_{|\underline{\mu}|\to\infty} |\underline{\mu}|^n |G(\underline{\mu})| = 0\right], \end{aligned} \tag{125}$$

gdzie norma wektora $\underline{\mu}$ może być obliczona jako

$$|\underline{\mu}|^2 = \sum_{\nu=1}^{n} (\mu_\nu)^2.$$

Przykładem ciągu η_k spełniającego warunki (124) może być ciąg liczbowy

$$\eta_k = \mu k^{-\nu}, \tag{126}$$

gdzie stałe μ i ν spełniają warunki:

$$[\mu > 0] \wedge [0 < \nu < \frac{1}{n}], \tag{127}$$

zaś jako przykłady funkcji jądra $G(\underline{\mu})$ można zaproponować:

$$G(\underline{\mu}) = \begin{cases} \eta, & \text{gdy} \quad |\underline{\mu}| \leq 1, \\ 0, & \text{gdy} \quad |\underline{\mu}| > 1, \end{cases} \tag{128}$$

gdzie η jest dowolną stałą, albo

$$G(\underline{\mu}) = (2\pi)^{-n/2} \exp\left(-\frac{1}{2}|\underline{\mu}|^2\right) \tag{129}$$

lub

$$G(\underline{\mu}) = 2^{-n} \exp\left(-\sum_{\nu=1}^{n} |\mu_\nu|\right), \tag{130}$$

albo

$$G(\underline{\mu}) = \prod_{\nu=1}^{n} (1 + |\mu_\nu|)^{-2}. \tag{131}$$

Przedstawimy algorytm omówionej metody, zakładając, że jądro oszacowania zadane jest funkcją

kernel(obj, sampl[k], seq[k]) – jądro oszacowania (123).

Dla skrócenia zapisu przyjmiemy dodatkowo, że określone są tablice:

seqn[1 . . num] – zawiera przeskalowane współczynniki ciągu uzbieżniającego $1/(\eta_k)^n$,

numi[1 . . numclass] – liczebność podzbiorów klas w ciągu uczącym N^i,

den[1 . . numclass] – wartości gęstości prawdopodobieństwa $p(\underline{x}/i)$ w punkcie $\underline{x}$.

```
begin
    den := 0 ;      {wyzerowanie całej tablicy}
    for k := 1 to num do
      begin
        i := sampl [k][dim + 1];
```

```
            den [i] := den [i] + seqn [k] * kernel(obj,sampl[k],seq[k]);
          end
          den [i] := den[i] * (1.0/numi[i]);
        for i := 1 to numclass do
          fun[i] := log (den[i])) + log(Prob[i]);
        rec := pointmax(fun);
    end
```

Wadą rozważanego podejścia jest konieczność utrzymywania w pamięci całego ciągu uczącego U, niezbędnego do efektywnego obliczenia $p(\underline{x}/i)$ według wzoru (123). Istnieje odmiana omówionej metody, pozwalająca na oszczędniejsze gospodarowanie pamięcią, opisana w pracy [9].

8.8. Algorytm *LI* jako szczególny przypadek metod probabilistycznych

Interesujące własności ma algorytm LI([12]) opracowany([13]) na podstawie nieparametrycznej estymacji funkcji gęstości prawdopodobieństwa $P(\underline{x}/i)$. Koncepcja tego algorytmu jest intuicyjnie prosta, jeśli jej wyjaśnienie zacząć od przypadku jednowymiarowej przestrzeni cech X ($n = 1$). W takim przypadku punkty ciągu uczącego U dzielą przestrzeń X na przedziały Δ, przy czym dla każdej klasy i (dla każdego podzbioru ciągu uczącego U^i) przedziały te (oznaczane Δ^i) inaczej się układają. Nieznany obiekt $\underline{x}$ trafia do jednego z tych przedziałów – oczywiście dla każdej klasy jest to *inny* przedział $\Delta^i(\underline{x})$. Istota algorytmu LI polega na tym, że należy wskazać jako poprawne rozpoznanie ten numer klasy i, któremu odpowiada *najmniejszy* przedział $\Delta^i(\underline{x})$. Funkcje przynależności są więc *odwrotnościami* długości odpowiednich przedziałów:

$$C^i(\underline{x}) = \frac{1}{\Delta^i(\underline{x})}.$$

([12]) Nazwa algorytmu pochodzi od słów „least interval" – najmniejszy przedział, co znajduje uzasadnienie w opisie metody.

([13]) Twórcą algorytmu LI jest prof. Zdzisław Bubnicki z Politechniki Wrocławskiej. Warto to podkreślić, ponieważ jest to najbardziej znaczący sukces Polaka w dziedzinie teorii rozpoznawania obrazów.

Dla przypadku $n > 1$ zasada ta musi być oczywiście zmodyfikowana w taki sposób, aby uwzględniać przedziały $\Delta^i(x_j)$ na wszystkich osiach x_j. Odpowiedni wzór, doprowadzony do formy użytecznej w praktyce ma postać:

$$C^i(\underline{x}) = p^i \frac{N^i}{(N^i - 1)^n \prod_{j=1}^{n} \Delta^i(x_j)}, \tag{132}$$

gdzie oznaczenia N^i oraz p^i były już używane, zaś kluczowe dla tego algorytmu długości przedziałów $\Delta^i(x_j)$ są wyznaczane w następujący sposob. Dokonywane jest porządkowanie każdej cechy x_j w każdym podciągu uczącym U^i, tak aby

$$\forall_{i \in I} \left[\forall_{j \in [1,n]} \left[\tilde{x}_j^{i,\nu} \leq \tilde{x}_j^{i,\nu} \right] \right], \tag{133}$$

gdzie ν jest nowym numerem porządkowym składowej $\tilde{x}_j^i$ będącej elementem wektora $\underline{\tilde{x}}^i \in U^i$. Wówczas

$$\Delta^i(x_j) = \tilde{x}_j^{i,k+1} - \tilde{x}_j^{i,k}, \tag{134}$$

gdzie k jest tak dobrane, aby składowa x_j wektora cech $\underline{x}$ rozpoznawanego obiektu $d \in D$ spełniała nierówność:

$$\tilde{x}_j^{i,k} < x_j < \tilde{x}_j^{i,k+1}. \tag{135}$$

Można na zasadzie umowy wprowadzić

$$\forall_{i \in I} \left[\forall_{j \in [1,n]} \left[x_j^{i,0} \equiv -\infty \wedge x_j^{i,N^i+1} \equiv +\infty \right] \right], \tag{136}$$

obejmując w ten sposób przypadek, kiedy tylko jedna z nierówności (135) jest spełniona, gdyż współrzędna rozpoznawanego obiektu $\underline{x}$ leży poza przedziałem wyznaczonym przez wartości odpowiednich współrzędnych obserwowanych w ciągu uczącym dla i-tej klasy.

Przykład. W pracy L. Kota badano poprawność rozpoznawania głosek szumowych różnymi metodami; wśród nich znalazły się wszystkie omawiane metody. Porównajmy zestawione w tabeli 8.2 wyniki rozpoznawania uzyskane kilkoma

metodami dla dwóch głosek([14]): łatwej do rozpoznawania głoski **w** oraz trudnej do rozpoznawania głoski **sz**.

Tabela 8.2. Poprawność rozpoznawania spółgłosek

Metoda	Poprawne rozpoznanie głoski (w procentach)	
	sz	**w**
Parzena	40,0	84,6
LI	20,0	80,8
NN	80,0	100,0
αNN	60,0	100,0
$j_N NN$	20,0	100,0
NM	60,0	96,2

Jak wynika z tych badań (oraz wielu innych przykładów) metoda Parzena daje istotnie gorsze wyniki rozpoznawania, niż metody minimalnoodległościowe, zaś algorytm *LI*, przy wszystkich swoich zaletach teoretycznych, także nie zawsze potwierdza swoją użyteczność – szczególnie w przypadku obrazów trudnych do rozpoznawania.

Algorytm opisanej metody wygodnie będzie przedstawić przy założeniu, że zbiór uczący, reprezentowany dotychczas w programach przez tablicę **sampl** zastąpiony zostanie tablicą

sampli [1 .. numclass][1 .. dim][1 .. maxnum] – ciąg uczący podzielony na klasy,

maxnum zastępuje tablica **numi[i]**, a używana już poprzednio (rozdz. 4) funkcja **sort** dopełnia liczby potrzebnych definicji.

```
begin
    for i := 1 to numclass do
    begin
      fun[i] := 1;
      for j := 1 to dim do
      begin
        sort(sampli [i][j]);    {uporządkowanie w klasie i od
                                 najmniejszej do największej wartości x^i
        k := 1;
```

([14]) Obie głoski podano w zapisie ortograficznym, a nie w transkrypcji fonematycznej.

```
            while obj[j] > sampli[i][j][k] and k <= numi[i] do
              k := k + 1;
            if k = 1 or k > numi[i]
              then
                begin
                  fun[i] := 0;
                  go to nextclass;
                end
              else
              fun[i] := fun[i] * (numi[i] - 1)*
                  (sampli[i][k][j]-sampli[i][k-1][j]);
            end
          fun[i] := Prawd[i] * numi[i]/fun[i];
          nextclass:;
        end
        rec := pointmax(fun);
    end
```

8.9. Rozpoznawanie etapowe

Wzmiankowane wielokrotnie metody rozpoznawania etapowego, zwane też metodami *sekwencyjnymi* (od klasycznej teorii *analizy sekwencyjnej* stosowanej przy statystycznym opracowywaniu wyników badań empirycznych – por. [40]) należą także do grupy omawianych tu metod probabilistycznych. Punktem wyjścia przy budowie tych metod jest test Walda([16]). Nadaje się on do bezpośredniego wykorzystania w zadaniu rozponawania etapowego w przypadku dychotomii (rozpoznawania dwóch klas, $L = 2$).

Załóżmy ponadto, że gęstość prawdopodobieństwa $P_z(\underline{x}/i)$ może być wyznaczona w kolejnych krokach (etapach procesu rozpoznawania) $z = 1, 2, \ldots, z_{max} < n$ przy znajomości jedynie niektórych składowych $x_\nu \in \underline{x}$.

Ma to miejsce na przykład przy niezależnych składowych x_ν, kiedy to w każdym kroku z można wybrać dowolny podzbiór $\tilde{x} \in 2^X$ (przy

([16]) W literaturze test ten oznaczany bywa symbolem WSPRT (Wald's Sequential Probability Ratio Test) – jakkolwiek podstawowe prace matematyczne A. Walda, dość stare, ale wciąż inspirujące badaczy (główna mongrafia pochodzi z 1947 roku!) opierały się na wcześniejszych badaniach H.F. Dodge'a i H.G. Romiga.

założeniu, że $\#\tilde{x}_1 < \#\tilde{x}_2 < \cdots < \#\tilde{x}_{z_{max}} = n$) i obliczać wartość $P_z(\underline{x}/i)$ ze wzoru:

$$P_z(\underline{x}/i) = \prod_{x_\nu \in \tilde{x}_z} p(x_\nu/i),$$

który warto porównać z analogiczną formułą (112). Załóżmy także znajomość dopuszczalnego prawdopodobieństwa błędów $e_{\mu\nu}$ dla wszystkich μ oraz $\nu = 1, 2, \ldots, L$. Przy zadaniu dychotomizacji ($L = 2$) z testu Walda wynika potrzeba oszacowania dwóch wartości, oznaczanych tutaj([17]) T_d i T_g:

$$T_g = \frac{1 - e_{21}}{e_{12}}, \qquad T_d = \frac{e_{21}}{1 - e_{12}}.$$

Podstawą do podjęcia decyzji jest stosunek prawdopodobieństw (porównaj wzór (97) pełniący w tej metodzie rolę funkcji rozdzielającej, zamiast używanej dotychczas funkcji przynależności (por. Dodatek 2):

$$C_z^{12}(\underline{x}) = \frac{P_z(\underline{x}/1)}{P_z(\underline{x}/2)}.$$

Na podstawie wartości funkcji $C_z^{12}(\underline{x})$ można ustalić właściwe rozpoznanie, dobierając konkretną postać odwzorowania F^e:

$$F^e\left[C_z^{12}(\underline{x})\right] = \begin{cases} 1, & \text{gdy} \quad C_z^{12}(\underline{x}) \geq T_g, \\ i_e, & \text{gdy} \quad T_g > C_z^{12}(\underline{x}) > T_d, \\ 2, & \text{gdy} \quad C_z^{12}(\underline{x}) \leq T_d. \end{cases}$$

Postać tej formuły w pełni uzasadnia intuicyjne przekonanie, jakie wiąże się z metodami rozpoznawania etapowego (sekwencyjnego): jeśli na etapie z pomierzono wystarczający zbiór cech $\tilde{x}_z$, to wówczas przekroczona zostaje jedna z granic danych parametrami T_d lub T_g i możliwe jest podjęcie jednoznacznej decyzji. Jeśli natomiast wartość funkcji rozdzielającej spełnia warunek $T_g > C_z^{12}(\underline{x}) > T_d$, to wówczas brak podstaw zarówno do uznania, że mamy do czynienia z klasą $i = 1$, jak do przyjęcia hipotezy alternatywnej $i = 2$. Należy wówczas przejść do kolejnego kroku $z + 1$ i uzupełnić zbiór cech o dodatkowe pomiary. Utworzony w ten sposób zbiór

([17]) W teście WSPRT używane są zwyczajowo oznaczenia A i B, jednak te symbole zostały w tej książce zarezerwowane do innych celów.

$\tilde{x}_{z+1}$ może pozwolić na dokładniejsze wyznaczenie wartości $C^{12}_{z+1}(\underline{x})$ i może doprowadzić do jednoznacznej decyzji. Jeśli nadal $F^e\left[C^{12}_z(x)\right] = i_e$, należy przejść do kroku $z+2$ itd. Procedura ta musi doprowadzić do jednej z dwóch możliwych sytuacji: albo na pewnym etapie z_i stanie się możliwe podjęcie jednoznacznego rozpoznania i, albo osiągnięty zostanie limit z_{max} bez możliwości podjęcia takiego rozpoznania. W tym drugim przypadku jedyną możliwą decyzją będzie definitywna odmowa rozpoznania: $i = i_o$.

Opisany sposób postępowania jest łatwy do przeprowadzenia dla zadania dychotomii ($L = 2$). Bardziej złożona sytuacja powstaje dla $L > 2$. Tu w ogóle nie jest możliwe podjęcie pozytywnej decyzji: jedyne, co można robić, to kolejna eliminacja w poszczególnych krokach z klas, do których rozważany obiekt $\underline{x}$ na pewno nie należy. Wykorzystuje się tu także wartości graniczne T^i (oddzielne dla każdej klasy i), obliczane ze wzoru:

$$T^i = \frac{1 - e_{ii}}{\left[\prod_{\nu=1}^{L}(1 - e_{i\nu})\right]}$$

oraz funkcje przynależności

$$C^i_z = \frac{P_z(\underline{x}/i)}{\left[\prod_{\nu=1}^{L} P_z(\underline{x}/\nu)\right]}.$$

Oczywiście w obydwu przytoczonych wzorach $i = 1, 2, \ldots, L$ oraz $z = 1, 2, \ldots, z_{max}$. Podejmowanie decyzji F^e polega na wprowadzeniu ciągu podzbiorów $I_0, I_1, I_2, \ldots, I_z, \ldots, I_{z_{max}} \forall_z (I \subseteq I)$ o malejącej liczebności

$$L = \#I_0 > \#I_1 > \#I_2 > \ldots > \#I_z > \ldots > \#I_{z_{max}} \geq 1,$$

tworzonych według *iteracyjnej* zasady:

$$I_0 = I, \qquad I_z = I_{z-1} - \tilde{I}_z,$$

gdzie symbol „−" oznacza różnicę zbiorów. Funkcjonowanie algorytmu ropoznającego polega na usuwaniu ze zbioru I_{z-1} wszystkich elementów zbioru $\tilde{I}_z$ zdefiniowanego w sposób następujący:

$$\tilde{I}_z = \{i : C^i_z < T^i\};$$

zbiór usuwanych elemntów $\tilde{I}_z$ może być pusty, jeśli w danym kroku z nie uda się dla żadnej klasy i ustalić nierówności $C_z^i < T^i$ będącej podstawą do jej wykluczenia z dalszych rozważań. W takim przypadku trzeba powtarzać próby eliminacji dla dalszych wartości z, wprowadzając do rozważań kolejne dalsze cechy x_ν. Może się również zdarzyć, że $\tilde{I}_z = I_{z-1}$, co oznacza, że po usunięciu wszystkich spełniających warunki klas pozostaje zbiór pusty ($I_z = \emptyset$). Decyzja podejmowana na kroku z zależy głównie od tego, czy liczba klas $\#I_z$ pozostających po dokonanej eliminacji na tym kroku jest mniejsza czy większa od ustalonej wartości ε (zwykle przyjmuje się $\varepsilon = 1$).

$$F^e\left[C_z^1(\underline{x}), C_z^2(\underline{x}), \ldots, C_z^L(\underline{x})\right] = \begin{cases} I_z, & \text{gdy} \quad \#I_z \leq \varepsilon, \\ i_e, & \text{gdy} \quad \#I_z > \varepsilon, \\ i_o, & \text{gdy} \quad \#I_z = 0. \end{cases}$$

Konsekwentne stosowanie podanej reguły jest podstawą funkcjonowania omówionego tu algorytmu. Szczegóły tego algorytmu oraz jego uzasadnienie znaleźć można głównie w pracach K.S. Fu [41, 42]. Można wykazać (chociaż nie jest to oczywiste na pierwszy rzut oka), że opisana ogólna metoda rozpoznawania jest równoważna wprowadzonemu na początku podrozdziału testowi Walda (WSPRT) dla przypadku $L = 2$.

Wprowadzając algorytm metody rozpoznawania etapowego wprowadzamy nowe obiekty, jakimi są tablice:

pr[1 . . numclass] – wartości prawdopodobieństwa $P_z(\underline{x}/i)$ dla poszczególnych klas (wyznaczne poza prezentowanym algorytmem po każdorazowym pomiarze wartości kolejnej cechy),

T[1 . . numclass] – wartości progowe (T^i),

acc[1 . . numclass] – wartości logiczne, określające które klasy nie zostały jeszcze wykluczone (zakłada się, że na początku *wszystkie* elementy tej tablicy mają wartość **true**).

```
begin
    prod := 1;      {wyznaczanie mianownika}
    for i := 1 to numclass do
      prod := prod * pr[i];
    count := 0;
    for i:= 1 to numclass do
```

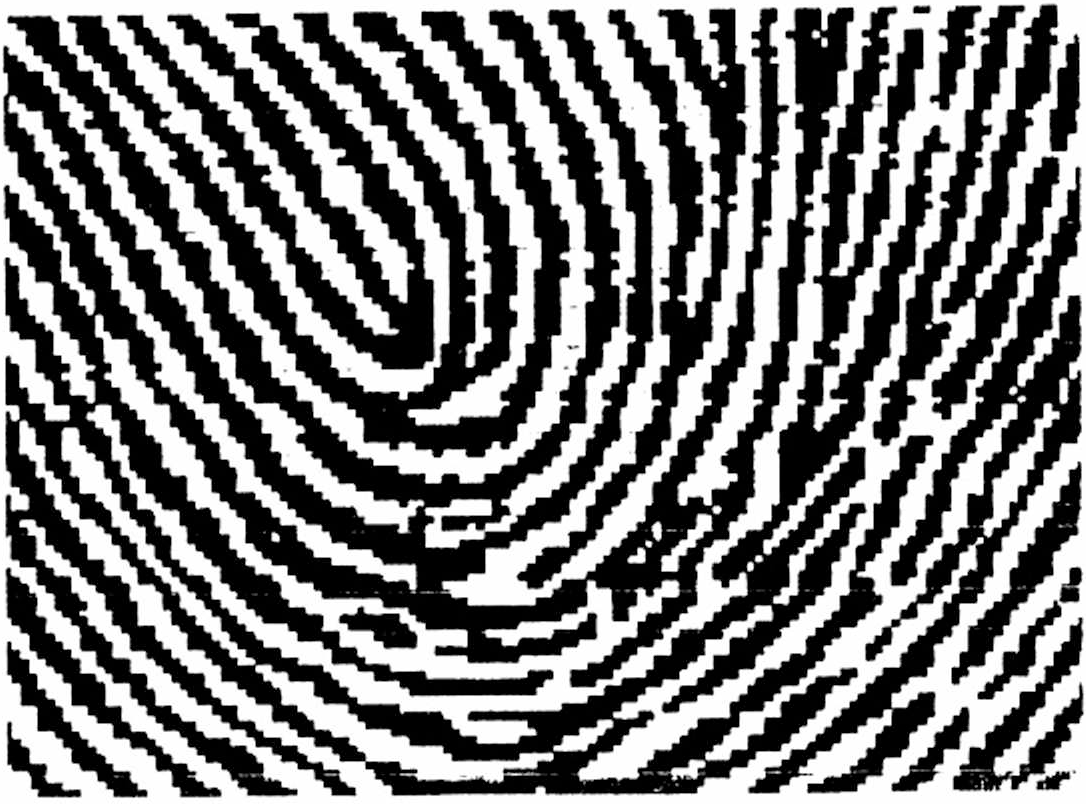
```
        begin
          if pr[i]/prod > T[i] then acc [i] := false;
          if acc[i] then
          begin
            count := count + 1;
            rec := i;
          end
        end
        if count = 0 then rec := 0;
        if count > 1 then rec := ALERT;
    end
```

9. WPROWADZENIE DO SYNTAKTYCZNEGO ROZPOZNAWANIA OBRAZÓW

W połowie lat sześćdziesiątych okazało się, że nawet zaawansowane metody matematyczne, przedstawione w poprzednich rozdziałach, są w pewnych przypadkach mało skuteczne. Szczególnie dotyczyło to sytuacji, w których rozważane obrazy były bardzo złożone, lub liczba klas była bardzo duża, co z kolei powodowało znaczne zwiększenie się wymiaru przestrzeni cech użytych do opisu wzorców. Klasycznym przykładem takiego problemu jest identyfikacja odcisków palców (rys. 9.1). Odpowiedzią na pojawienie się tego rodzaju problemów było powstanie syntaktycznego podejścia do rozpoznawania obrazów wykorzystującego metody lingwistyki matematycznej.

Przy podejściu syntaktycznym, złożony obraz jest dzielony na prostsze podobrazy, które próbuje się rozpoznawać metodami całościowymi, traktując je jako niezależne. W przypadku, gdy podobrazy te nadal charakteryzują się dużą złożonością, operacja podziału jest kontynuowana tak długo, aż otrzymamy składowe pierwotne obrazu (ang. picture primitives). Składowe pierwotne obrazu są takimi niepodzielnymi elementami obrazu, o których zakładamy, że istnieją i, co więcej, zakładamy, że dla danej klasy obrazów potrafimy je a prori zdefiniować, Opisany sposób postępowania przedstawiono graficznie na rysunku 9.2.

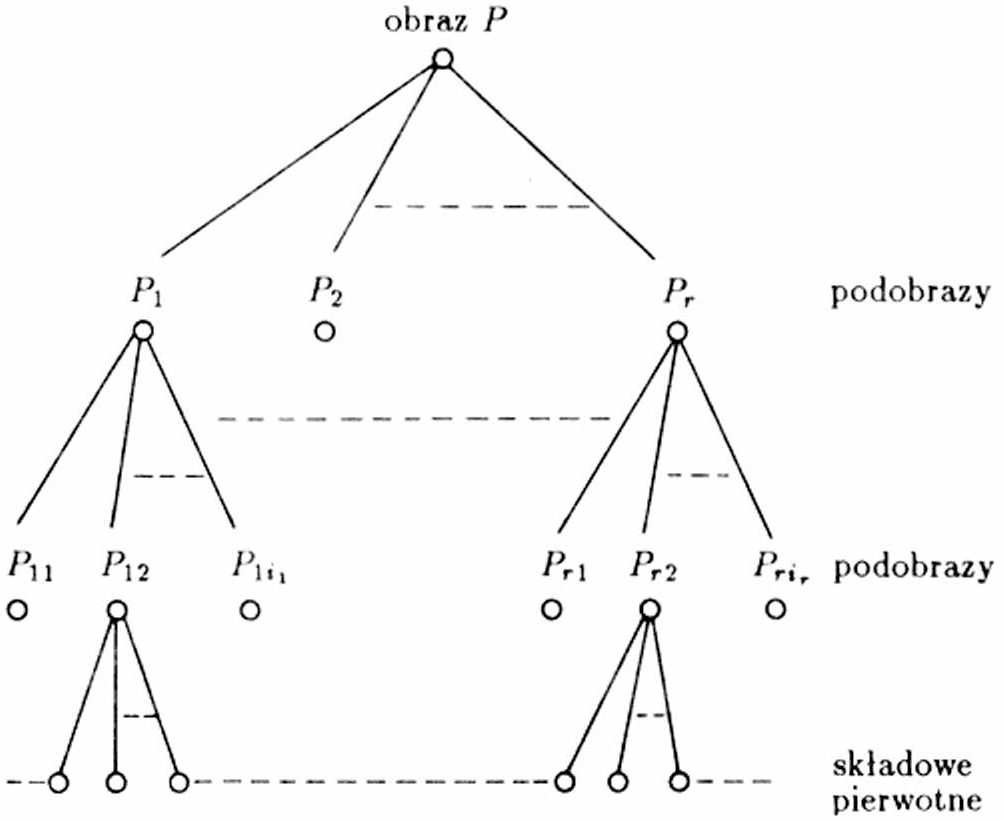
Jak więc widać, w przypadku metod syntaktycznych, przed przystąpieniem do właściwego rozpoznawania należy wyodrębnić składowe pierwotne oraz relacje, jakie zachodzą między nimi. Najczęściej spotykanym sposobem wyodrębniania składowych pierwotnych jest segmentacja obrazu, np. metodami wykrywania krawędzi za pomocą operatorów gradientowych lub operatorów dopasowania wzorców. Omówienie tych (oraz innych) metod wyodrębniania (rozpoznawania) składowych pierwotnych, Czytelnik może



Rys. 9.1. Obraz odcisku palca wprowadzony do komputera i poddawany rozpoznawaniu może być przykładem złożonego obiektu rozpoznawania, do którego identyfikacji nadają się metody syntaktyczne

znaleźć między innymi w publikacjach [53], [54] i [55]. Identyfikacja relacji jest niezmiernie ważnym etapem, gdyż, jak już wspomnieliśmy, rozpoznając podobrazy (a więc i składowe pierwotne) traktujemy je jako niezależne obrazy. Zatem prawidłowe złożenie całego obrazu z podobrazów warunkuje prawidłowe jego rozpoznanie. Waga tego etapu ma swoje odzwierciedlenie w fakcie, iż podstawowej klasyfikacji syntaktycznych metod rozpoznawania obrazów dokonuje się właśnie ze względu na charakter tych relacji (por. rozdz. 3). Dlatego też, zanim przedstawimy ideę mechanizmu rozpoznającego obrazy strukturalne, omówimy krótko trzy główne metody formalne używane do reprezentacji obrazów tego rodzaju, ilustrując je prostymi przykładami.

Najwcześniej powstałą grupą metod są *metody ciągowe*. Jak już sama nazwa wskazuje, obraz jest reprezentowany w tym przypadku przez *ciąg*, co oznacza, że wyróżniamy jeden rodzaj relacji, jaki może zachodzić po-

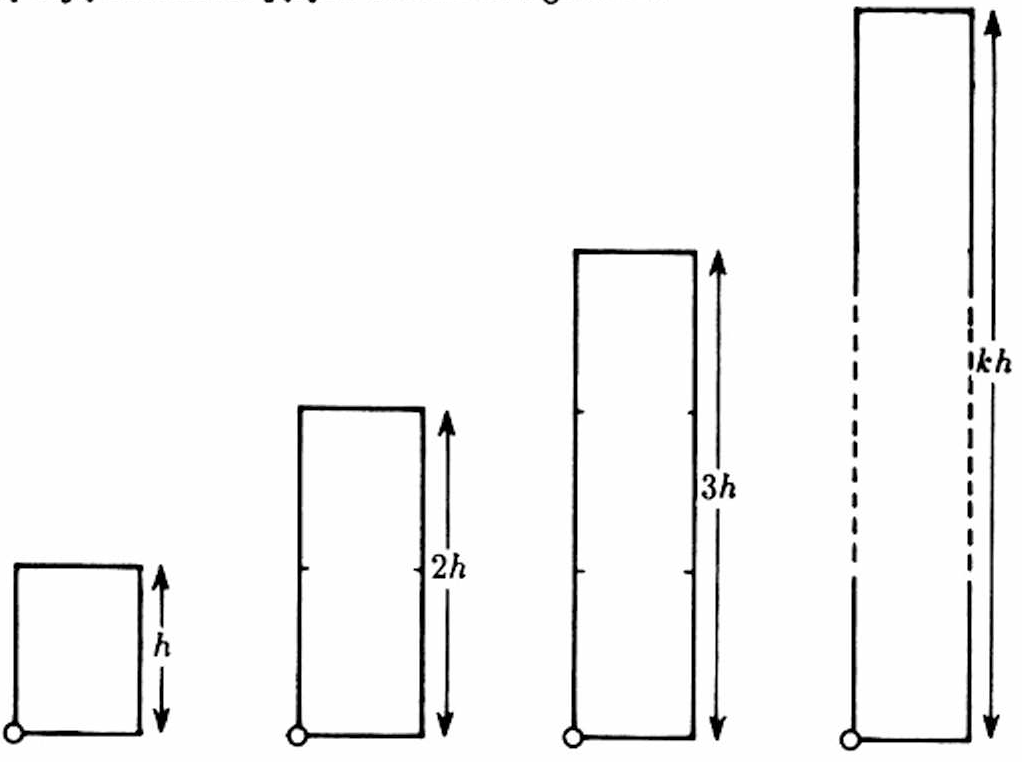


Rys. 9.2. Graf dekompozycji obrazu na elementy składowe

między składowymi pierwotnymi. Relacja ta to *konkatenacja* (doklejenie) kolejnego elementu. Rozważmy przykładowo zbiór obrazów przedstawiony na rysunku 9.3. Jeśli założymy, że zbiór składowych pierwotnych jest określony jak na rysunku 9.4, to obraz pierwszy możemy opisać ciągiem *abcd* (startując z punktu zaznaczonego kółkiem i poruszając się wzdłuż konturu zgodnie z ruchem wskazówek zegara), obraz drugi – $aabccd = a^2bc^2d$, trzeci – $aaabcccd = a^3bc^3d$. Ogólnie klasę takich obrazów (zauważmy, że jest to zbiór nieskończony!) opiszemy prostą formułą w postaci ciągu a^kbc^kd, $k > 0$. Tak zdefiniowana formuła definiuje zbiór prostokątów, których jeden bok ma długość jednostkową. Ogólniejszą klasę – zbiór prostokątów opiszemy formułą $a^kb^mc^kd^m$, $k, m > 0$, a klasę kwadratów opiszemy ciągiem $a^kb^kc^kd^k$.

Języki ciągowe służą do opisu i rozpoznawania pojedynczych (niejednokrotnie bardzo złożonych) obiektów obrazu. Klasycznymi przykładami takich języków są: języki oparte na kodach łańcuchowych (Freemana [14,15]),

języki opisu obrazów (Shawa [16]) i języki opisu cech kształtów (Jakubowskiego [17,18]). Zostaną one omówione w następnym rozdziale. Języki omawianej tu klasy, nawet o tak dużej mocy opisowej jak języki opisu cech kształtów, nie dają możliwości opisu obrazów wieloobiektowych. Możliwość taką dają natomiast języki drzewowe i grafowe.



Rys. 9.3. Zbiór obrazów poddanych opisowi strukturalnemu

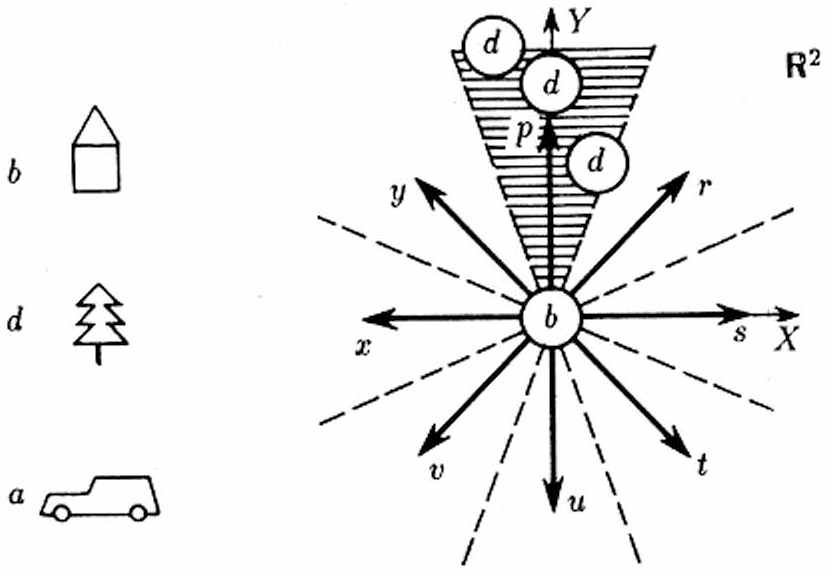
Rys. 9.4. Zbiór składowych pierwotnych do opisu obrazów z rys. 9.3

Języki drzewowe charakteryzują się szerokim spektrum zastosowań. Pełny przegląd tych zastosowań znajduje się między innymi w [19]. W rozdziale 10, poświęconym w całości metodom drzewowym, ich prezentację zilustrujemy dwiema, najczęściej spotykanymi w literaturze, aplikacjami: analizą scen (ang. *scene analysis)* i analizą faktur (ang. *texture analysis).*

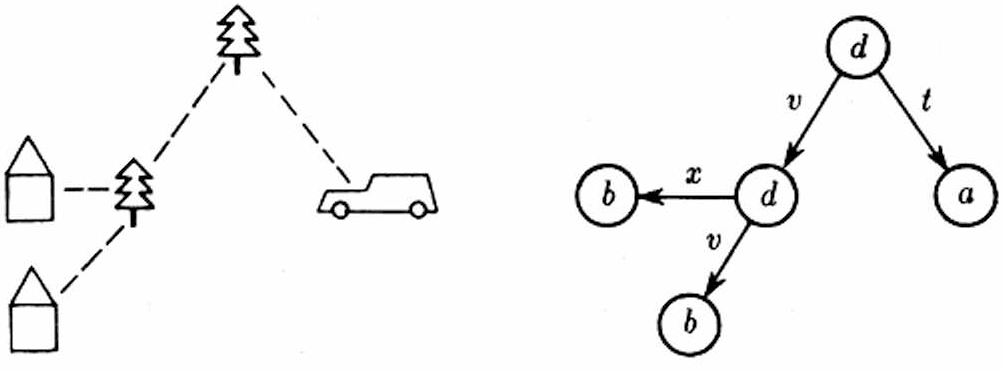
Analizą sceny nazywamy proces rozpoznawania trójwymiarowego obrazu, w którym wyodrębniono i zidentyfikowano *obiekty* składające się na ten obraz oraz opisano *relacje*, jakie zachodzą pomiędzy tymi obiektami. W tym przypadku wyodrębniamy dwa zbiory składowych pierwotnych. Jeden służy do opisu obiektów, z jakich złożony jest obraz, drugi – opisuje relacje (najczęściej przestrzenne), jakie mogą zachodzić pomiędzy obiektami. Przykładowo przyjmijmy, że oba zbiory są przedstawione na rysunku 9.5. Zbiór opisujący *obiekty* obrazu zawiera składowe pierwotne: budynek (b), drzewo (d), automobil (a). Zbiór określający *relacje*, jakie mogą zachodzić pomiędzy obiektami obrazu dzieli płaszczyznę, w której znajdują się obiekty na osiem podobszarów. Jeśli, na przykład, obiekt typu drzewo (d), dla którego będziemy określać relację względem drugiego obiektu typu budynek (b) znajdującego się w początku układu współrzędnych XOY, będzie umiejscowiony w zacienionym obszarze, to relację określimy jako p. Przy tak zdefiniowanych zbiorach, możemy reprezentować scenę przez drzewo([1]) tak, jak jest to pokazano na rysunku 9.6.

Pojęcia *faktury* będziemy używać do określenia własności powtarzającego się motywu (wzorca), wypełniającego całą rozważaną płaszczyznę. Analiza faktury może być celem samodzielnym, na przykład w przypadku analizy obrazów, na których obiekty w ogóle nie występują (na przykład w badaniach metalograficznych lub ocenie preparatów geologicznych), a także jest szczególnie ważna w przypadku rozpoznawania obrazów słabo wykontrastowanych, kiedy oddzielenie obiektów od tła jest trudne (na przykład zdjęcia satelitarne – rys. 9.7). Przy podejściu syntaktycznym, obraz jest dzielony na okienka o stałych rozmiarach, możliwie tak, aby w okienku znalazł się powtarzający się motyw. Następnie motyw ten jest

([1]) Drzewem będziemy nazywać spójny i acykliczny graf. Na rysunku 9.6 drzewo ma zaetykietowane wierzchołki oraz zaetykietowane i skierowane krawędzie. Przypomnijmy jeszcze, że wierzchołek, od którego drzewo t bierze początek nazywamy korzeniem drzewa t (na rysunku 9.6 korzeniem drzewa t jest wierzchołek d), natomiast wierzchołki, z których nie wychodzą żadne krawędzie (na rysunku 9.6 – b, b, a) – nazywamy liśćmi. Zapisem drzewa t w postaci nawiasowej (prefiksowej) nazywamy opis skonstruowany rekurencyjnie w następujący sposób : $I_0 = k(I_1 \ldots I_n)$, gdzie k jest etykietą korzenia drzewa t, I_i jest zapisem w postaci nawiasowej drzewa, dla którego korzeniem jest i-ty bezpośredni następnik wierzchołka k. Przykładowo dla drzewa z rysunku 9.6 będzie to zapis $d(d(bb)a)$. Jeśli chcemy uwzględnić w takim zapisie etykiety krawędzi, to przed każdą etykietą wierzchołka dopisujemy etykietę krawędzi wchodzącej do tego wierzchołka. W naszym przykładzie otrzymamy wtedy: $d(vd(xbvb)ta)$. Wprowadzone powyżej (w intuicyjny sposób) pojęcia będziemy używać w rozdziale 10. Formalne definicje Czytelnik może znaleźć, przykładowo, w [20].



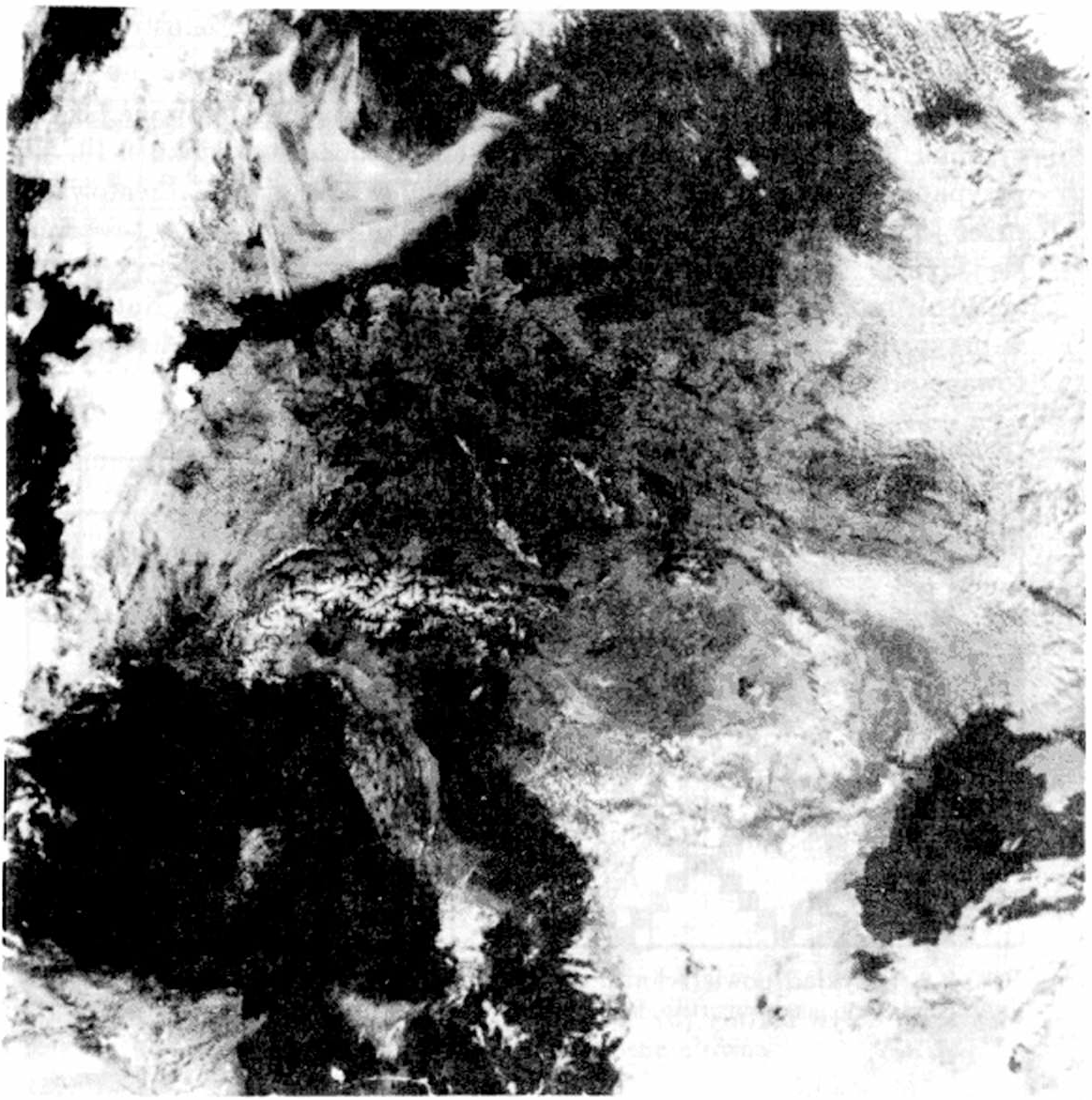
Rys. 9.5. Zbiór obiektów (po lewej stronie) i zbiór relacji (po prawej) używanych do strukturalnego opisu obrazów



Rys. 9.6. Prosta scena (po lewej stronie) i opis tej sceny za pomocą obiektów i relacji wprowadzonych na rys. 9.5

reprezentowany w postaci drzewa. Rozważmy przykładowo powierzchnię o fakturze przedstawionej na rysunku 9.8a.

Powtarzający się motyw, umieszczony w okienku 9×9 znajduje się na rysunku 9.8b, natomiast drzewo rozpinamy w okienku tak, jak to zaprezentowano na rysunku 9.9a (strzałką zaznaczono piksel odpowiadający korzeniowi drzewa). Przyjmując, że zbiór składowych pierwotnych złożony jest z czarnego piksela opisanego przez 1 i białego – opisanego przez 0,

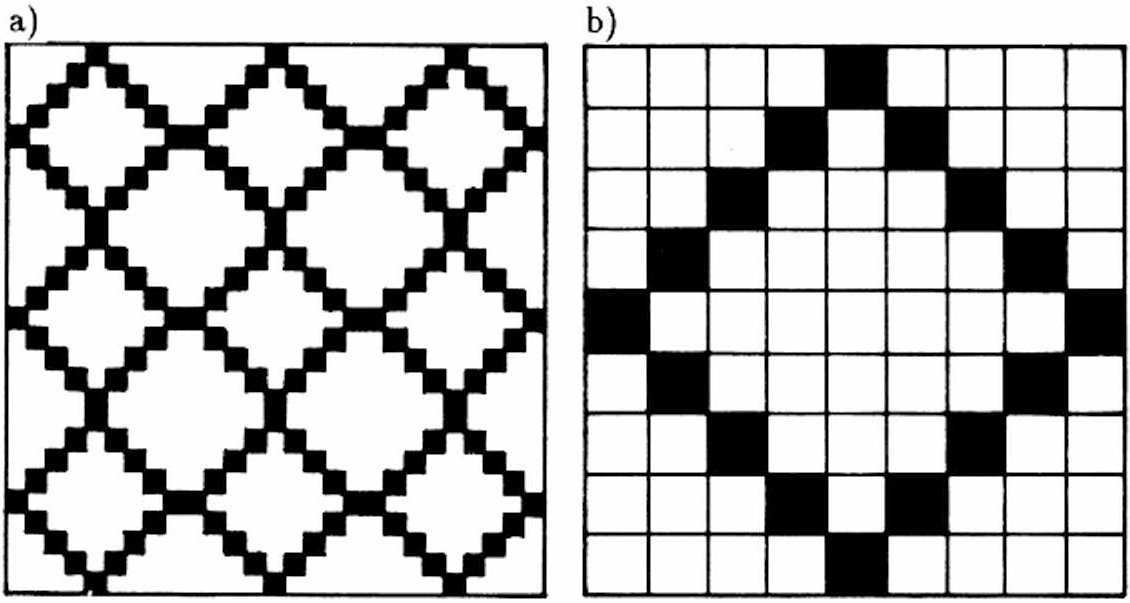


Rys. 9.7. Zdjęcia satelitarne, będące wyjątkowo częstym obiektem analizy i rozpoznawania komputerowego, stanowią przykład obrazów, przy rozpoznawaniu których wykorzystuje się analizę faktury

możemy reprezentować rozważany motyw przez drzewo znajdujące się na rysunku 9.9b.

Trzecią grupą rozważanych tu metod są *metody grafowe*. Użycie grafów do opisu złożonych dwu- i trójwymiarowych obrazów wieloobiektowych jest powszechnie spotykane w literaturze. Związane jest to z faktem, że graf jest mocniejszym formalizmem opisowym niż drzewo. Zilustrujmy to następującym przykładem.

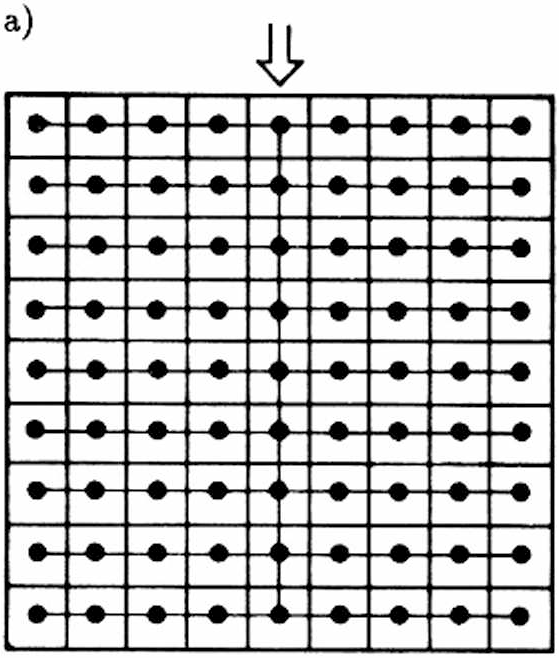
Niech obiekty sceny i relacje między nimi będą reprezentowane jak na rysunku 9.5. Rozważmy dwie sceny przedstawione na rysunku 9.10. W przypadku użycia reprezentacji drzewowej obie sceny są reprezentowane przez to samo drzewo przedstawione na rysunku 9.6, gdyż sceny te różnią się jedynie relacją pomiędzy górnym a dolnym budynkiem. Relacji tej, niestety, nie możemy określić, ponieważ w drzewie powstałby cykl. Natomiast w przypadku reprezentacji grafowej obrazy te są rozróżnialne i reprezentowane przez grafy znajdujące się na rysunku 9.11.



Rys. 9.8. Przykład powierzchni o wyraźnej fakturze (a) i powtarzający się, elementarny motyw faktury (b)

Po omówieniu problemu reprezentacji obrazu (kluczowego w przypadku metod syntaktycznych), przedstawimy teraz ideę *syntaktycznego rozpoznawania obrazów*. Podstawowym założeniem tych metod jest możliwość zdefiniowania mechanizmu generującego reprezentacje (ciągowe, drzewowe lub grafowe) rozważanych obrazów [19]. Tym mechanizmem jest *gramatyka* G

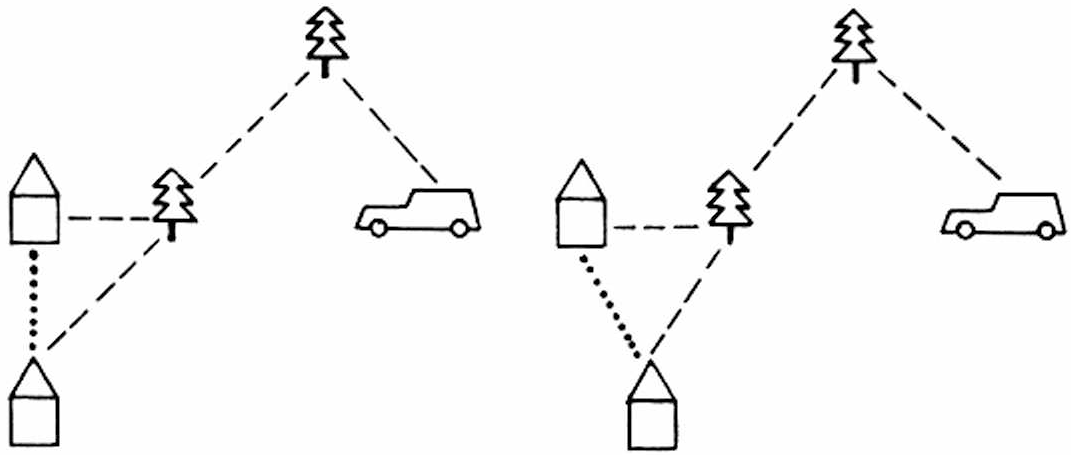
a)



b)

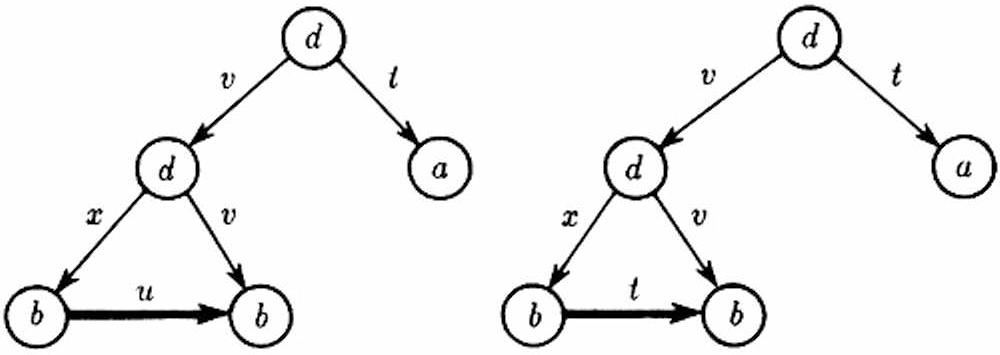
0 – 0 – 0 – 0 – 1 – 0 – 0 – 0 – 0
0 – 0 – 0 – 1 – 0 – 1 – 0 – 0 – 0
0 – 0 – 1 – 0 – 0 – 0 – 1 – 0 – 0
0 – 1 – 0 – 0 – 0 – 0 – 0 – 1 – 0
1 – 0 – 0 – 0 – 0 – 0 – 0 – 0 – 1
0 – 1 – 0 – 0 – 0 – 0 – 0 – 1 – 0
0 – 0 – 1 – 0 – 0 – 0 – 1 – 0 – 0
0 – 0 – 0 – 1 – 0 – 1 – 0 – 0 – 0
0 – 0 – 0 – 0 – 1 – 0 – 0 – 0 – 0

Rys. 9.9. Struktura drzewa , na którym rozpinana jest faktura z rys. 9.8a (a) i reprezentacja motywu faktury przez drzewo (b)



Rys. 9.10. Przykład dwóch scen, których opis strukturalny za pomocą drzewa jest identyczny, a których topografia jest znamiennie różna

(ciągowa, drzewowa lub grafowa), a zbiór wszystkich reprezentacji obrazów generowanych przez nią jest traktowany jako pewien *język*. Możemy zatem skonstruować automat $\mathfrak{A}$ rozpoznający elementy tego języka. Automat ten, a raczej jego programowa implementacja – analizator syntaktyczny (ang.



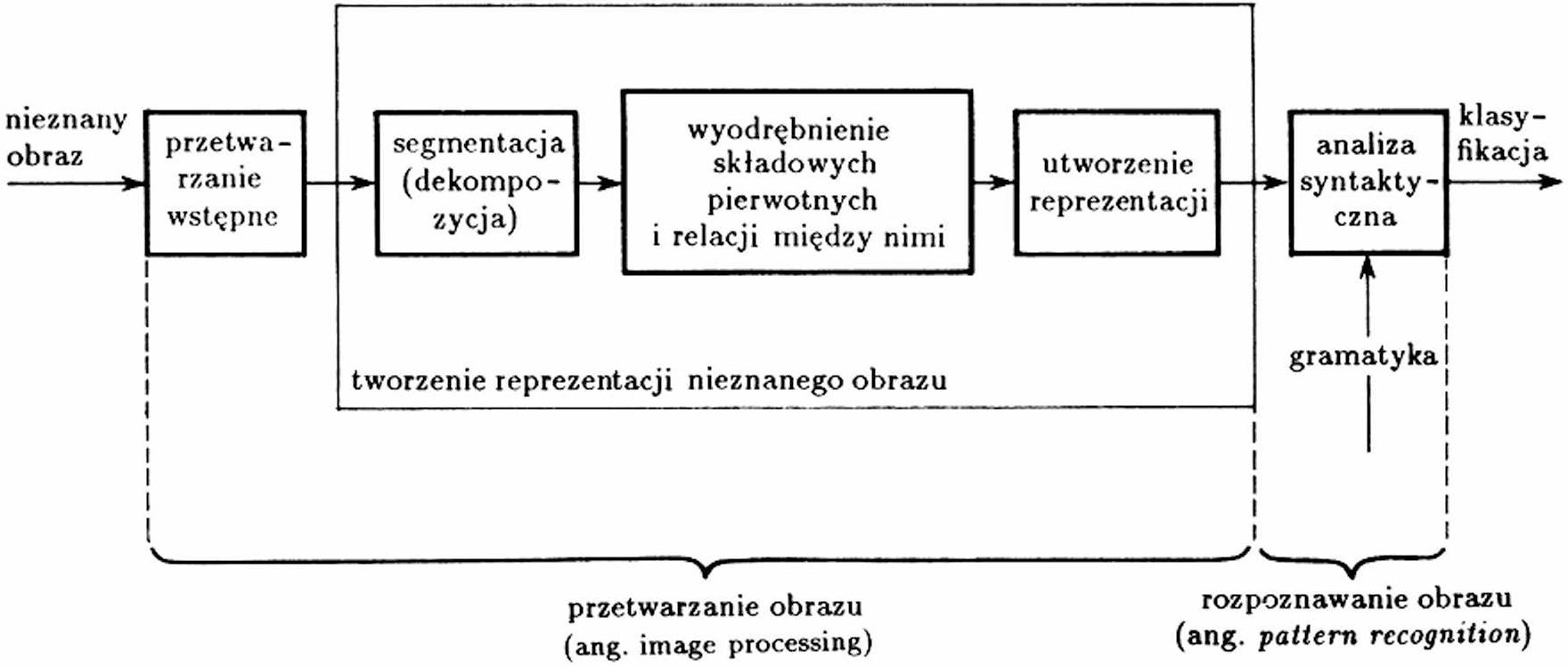
Rys. 9.11. Opis strukturalny scen z rys. 9.10 za pomoca grafów pozwala na wydobycie cech, ktore je różnią

parser) odpowiada procedurze rozpoznającej F jednym z dwóch następujących sposobów.

Niech zbiór obiektów D rozbija się na kolekcję L klas równoważności $\{D^i \mid i = 1, \ldots L\}$ nazywanych – jak wszędzie dotychczas – obrazami. Zdefiniujmy zbiór stanów końcowych automatu (patrz Dodatek 3) – $F_{\mathfrak{A}} = \{f_1, \ldots, f_L, err\}$ tak, że dla każdego obiektu $d \in D^i$, automat po przeanalizowaniu reprezentacji $\underline{x}$ obiektu d przejdzie w stan f_i. Tak więc rozpoznanie $i \in I$ obiektu $d \in D$ będzie w tym przypadku ustalane na podstawie identyfikacji stanu końcowego automatu $\mathfrak{A}$. Brak decyzji i_o odpowiada w automacie $\mathfrak{A}$ stanowi błędu err.

Drugim sposobem jest konstrukcja automatu z wyjściem $\mathfrak{A}_w$ (patrz definicja automatu $LL(1)$ w Dodatku 3). Automat taki, dokonując analizy reprezentacji x obiektu d, wypisuje w kolejnych krokach numery tych produkcji $\mathfrak{P}$ gramatyki $\mathfrak{G}$, których należałoby użyć, aby wygenerować reprezentację $\underline{x}$. Zbiór stanów końcowych automatu jest w tym przypadku dwuelementowy $F = \{acc, err\}$, gdzie acc oznacza stan akceptacji przeanalizowanej reprezentacji (w sensie jej przynależności do języka rozpoznawalnego przez automat $\mathfrak{A}_w$), czyli stan rozpoznania obiektu d, natomiast err oznacza (jak poprzednio) brak decyzji. Identyfikacji $i \in I$ odpowiedniej klasy D^i dokonujemy, przy tym podejściu, na podstawie otrzymanego na wyjściu ciągu numerów $\{\mu_i\}$ produkcji $\mathfrak{p}_\mu \in \mathfrak{P}$ (z każdą klasą $i \in I$ związany jest zbiór ciągów numerów $\{\mu_i\}$ produkcji $\mathfrak{P}$ gramatyki $\mathfrak{G}$).

Podsumowując, główną ideą syntaktycznego rozpoznawania obrazów jest reprezentowanie obiektu $d \in D$ przez ciąg (sznur), drzewo lub graf



Rys. 9.12. Schemat toru przetwarzania obiektów rozpoznawania (na ogół obrazów scen statycznych lub dynamicznych) pozwala wyróżnić zasadnicze etapy strukturalnego rozpoznawania obrazów

podobrazów, a następnie konstruowanie analizatora syntaktycznego $\mathfrak{A}$ dla gramatyki $\mathfrak{G}$ (ciągowej, drzewowej lub grafowej) generującej cały obiekt x z tych podobrazów. Ponieważ konstrukcja procedury rozpoznającej (automatu $\mathfrak{A}$) opiera się na definicji procedury generującej – gramatyki $\mathfrak{G}$, więc prezentując kolejne metody, będziemy najpierw przedstawiać gramatykę $\mathfrak{G}$, a później automat $\mathfrak{A}$.

Rozdział ten zakończymy przedstawieniem toru przetwarzania obiektów w syntaktycznym systemie rozpoznawania obrazów. Proces rozpoznawania, zilustrowany rysunkiem 9.12, składa się z dwóch części: przetwarzania wstępnego obrazu (ang. *image preprocessing)* oraz właściwego rozpoznawania (ang. *pattern recognition*). Przetwarzanie obrazu, odpowiadające odwzorowaniu $B : D \rightarrow X$ (recepcji) w metodach całościowych, jakkolwiek jest częścią systemu syntaktycznego rozpoznawania obrazów, to nie zawiera się w problematyce rozpoznawania obrazów i stanowi oddzielną dziedzinę badań. Właściwe rozpoznawanie obrazu, $\hat{A}$ mające charakter analizy syntaktycznej, prowadzi do identyfikacji obrazu jednym z dwóch opisanych sposobów.

10. METODY CIĄGOWE

10.1. Uwagi ogólne

W tym rozdziale omówimy trzy spośrod wielu znanych metod ciągowych rozpoznawania obrazów. Pierwsza z nich, oparta na *kodach łańcuchowych* Freemana (ang. Freeman *chain codes*) [14,15], jest jedną z najwcześniejszych syntaktycznych metod rozpoznawania obrazów. Charakteryzuje się ona prostotą reprezentacji $\underline{x}$ obiektu $d \in D$ oraz liniową złożonością procedury rozpoznającej, którą jest deterministyczny automat $\mathfrak{A}$ o skończonej liczbie stanów (języki oparte na kodach łańcuchowych są zwykle regularne). *Języki opisu obrazów* $\mathfrak{L}_{PDL}$ (ang. *picture description languages - PDL)* zostały pierwotnie zdefiniowane przez Shawa jako narzędzie opisu i rozpoznawania obrazów torów cząstek elementarnych w Stanford Linear Accelerator Center [16]. Obrazy takie były opisywane za pomocą zbioru skierowanych *składowych pierwotnych* oraz *operatorów* określających relacje wzajemnego położenia tych składowych. Języki opisu obrazów $\mathfrak{L}_{PDL}$ zwykle są generowane przez gramatyki bezkontekstowe $\mathfrak{G}_b$ (czasem z operatorem indeksowania). Oznacza to, że charakteryzują się one większą mocą opisową niż kody łańcuchowe. Z drugiej strony, powoduje to konieczność użycia automatu ze stosem $\mathfrak{A}_s$ jako procedury rozpoznającej $\hat{A}$. *Języki opisu cech kształtów* (Jakubowski) $\mathfrak{L}_{SFDL}$ (ang. *shape feature description languages - SFDL*) są obecnie (1990 rok) najmocniejszym formalizmem opisowym wśród języków ciągowych znanych z literatury [26]. Dzięki wielopoziomowej, hierarchicznej strukturze tych języków możliwa jest nie tylko klasyfikacja obiektu na podstawie opisu jego kształtu wyrażonego przez składowe pierwotne, ale również wydobywanie cech tego kształtu, tzn. „rozumienie" kształtu (w takim znaczeniu tego słowa, jakie jest przyjmowane w AI).

W kolejnych podrozdziałach przedstawimy te metody, prezentując: mechanizm generujący – gramatykę $\mathfrak{G}$, za pomocą której możemy zapamiętać w dynamiczny sposób wzorce $\underline{x}^k$ ciągu uczącego U, oraz procedurę analizującą i rozpoznającą – automat $\mathfrak{A}$ odpowiedniej klasy. Prezentację zilustrujemy przykładami rozpoznawania kilku zaledwie wzorców $\underline{x}^k$ klas D^i, gdyż wprowadzenie kilkudziesięciu (kilkuset, kilku tysięcy) wzorców (z takimi licznościami mamy do czynienia w praktyce) tylko dla ilustracji nie byłoby sensowne. Z drugiej jednak strony użycie kilku zaledwie wzorców nie pozwoli Czytelnikowi naocznie przekonać się, że użycie dynamicznej formuły generacji ciągu uczącego U – gramatyki $\mathfrak{G}$ jest znacznie mniej pamięciochłonne niż pamiętanie ciągu uczącego *explicite*, w przypadku gdy jest on bardzo liczny([1]).

10.2. Kody łańcuchowe Freemana

Metodę opartą na kodach łańcuchowych Freemana przedstawimy na przykładzie wzorców czterech drukowanych liter: **I, P, R, D**. Składowe pierwotne kodu Freemana przedstawiono na rysunku 10.1a, a reprezentacje rozważanych liter zbudowane na bazie kodu – na rysunku 10.1b. Jak zatem widać, kolejne litery możemy zapisać za pomocą następujących ciągów (startując z punktu zaznaczonego kropką i zaznaczając przez \$ – koniec ciągu).

I – 0000\$,

P – 000023456\$,

R – 00002345633\$,

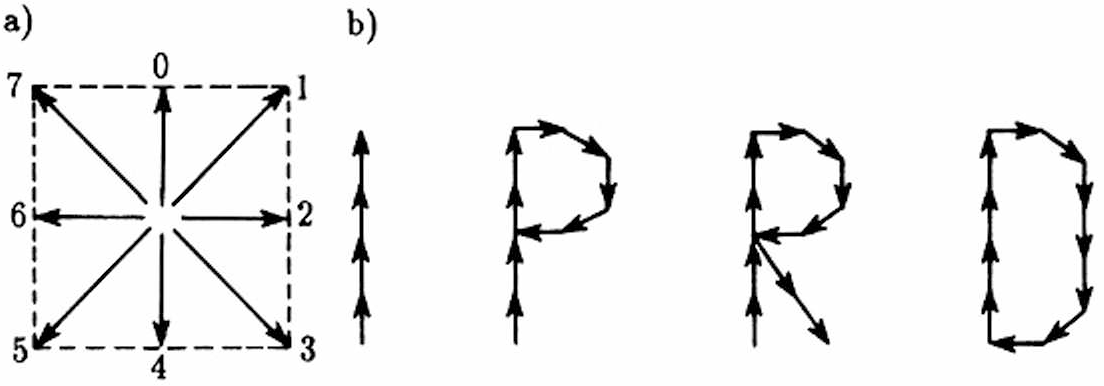
D – 00002344456\$.

Prawostronnie regularną gramatykę $\mathfrak{G}_r$ generującą te ciągi-reprezentacje, zdefiniujemy w następujący sposób:

$$\mathfrak{G}_r = (\Sigma_N, \Sigma_T, \mathfrak{P}, S),$$

gdzie zbiór $\mathfrak{P}$ omówimy z podziałem na grupy:

([1]) Pojęcia z teorii języków formalnych i automatów używane w niniejszym rozdziale zostały formalnie zdefiniowane w Dodatku 3.



Rys. 10.1. Składowe pierwotne kodu Freemana (a) i zbudowane za ich pomocą kontury kilku liter, traktowane jako wzorce do rozpoznania (b)

$\mathfrak{P}_1$: produkcje generujące I:

(1) $S \to OS_1$,

(2) $S_1 \to OS_2$,

(3) $S_2 \to OS_3$,

(4) $S_3 \to OS_4$,

(5) $S_4 \to \$$,

$\mathfrak{P}_2$: produkcje generujące P: (1), (2), (3), (4) oraz

(6) $S_4 \to 2S_5$,

(7) $S_5 \to 3S_6$,

(8) $S_6 \to 4S_7$,

(9) $S_7 \to 5S_8$,

(10) $S_8 \to 6S_9$,

(11) $S_9 \to \$$,

$\mathfrak{P}_3$: produkcje generujące R: (1), (2), (3), (4), (6), (7), (8), (9), (10) oraz

(12) $S_9 \to 3S_{10}$,

(13) $S_{10} \to 3S_{11}$,

(14) $S_{11} \to \$$,

$\mathfrak{P}_4$: produkcje generujące D: (1), (2), (3), (4), (6), (7), (8) oraz

(15) $S_7 \to 4S_{12}$,

(16) $S_{12} \to 4S_{13}$,

(17) $S_{13} \to 5S_{14}$,

(18) $S_{14} \to 6S_{15}$,

(19) $S_{15} \to \$$,

$\mathfrak{P} = \mathfrak{P}_1 \cup \mathfrak{P}_2 \cup \mathfrak{P}_3 \cup \mathfrak{P}_4$,

$\Sigma_N = \{S, S_i\}$, $i = 1, 2, \ldots, 15$,

$\Sigma_T = \{\$, 0, 2, 3, 4, 5, 6\}$.

Po skonstruowaniu gramatyki $\mathfrak{G}_r$, zbudujemy deterministyczny automat $\mathfrak{A}$ o skończonej liczbie stanów, rozpoznający poszczególne wzorce liter. Automat $\mathfrak{A}$ określamy według następujących reguł:

$$\mathfrak{A} = (\Sigma_T', Q, \delta, q_0, F),$$

gdzie (1) $\Sigma_T' := \Sigma_T$,

(2) $Q := \Sigma_N$,

(3) $q_0 := S$,

(4) $F := \emptyset$,

(5) jeśli produkcja postaci $A_1 \to aA_2 \in \mathfrak{P}$, $A_1, A_2 \in \Sigma_N$, $a \in \Sigma_T$, to $\delta(A_1, a) := A_2$,

(6) jeśli produkcja postaci $A_1 \to \$ \in \mathfrak{P}$, $\$ \in \Sigma_T$ jest produkcją kończącą generację obrazu Lit, to $\delta(A_1, \$) := Lit$ oraz $Q := Q \cup \{Lit\}$, $F := F \cup \{Lit\}$,

(7) dla pozostałych par postaci (A_1, a), które nie zostały rozpatrzone w punktach (5) i (6) $\delta(A_1, a) := err$, gdzie err jest stanem nierozpoznania obrazu (oczywiście $Q := Q \cup \{err\}$, $F := F \cup \{err\}$).

W naszym przykładzie mamy:

(1) $\Sigma_T' = \{\$, 0, 2, 3, 4, 5, 6\}$,

(2) $Q = \{S, S_i, I, P, R, D, err\}$, $i = 1, 2, \ldots, 15$,

(3) $q_0 = S$,

(4) $F = \{I, P, R, D, err\}$,

(5) $\delta(S,0) = S_1$, $\delta(S_4,2) = S_5$, $\delta(S_9,3) = S_{10}$, $\delta(S_7,4) = S_{12}$,
$\delta(S_1,0) = S_2$, $\delta(S_5,3) = S_6$, $\delta(S_{10},3) = S_{11}$, $\delta(S_{12},4) = S_{13}$,
$\delta(S_2,0) = S_3$, $\delta(S_6,4) = S_7$, $\delta(S_{11},\$) = R$, $\delta(S_{13},5) = S_{14}$,
$\delta(S_3,0) = S_4$, $\delta(S_7,5) = S_8$, $\delta(S_{14},6) = S_{15}$,
$\delta(S_4,\$) = I$, $\delta(S_8,6) = S_9$, $\delta(S_{15},\$) = D$,
$\delta(S_9,\$) = P$,

natomiast dla pozostałych argumentów funkcja δ przyjmuje wartość *err*.

Dokonajmy rozpoznania litery **D** (opisanej przez ciąg 000023444456$) za pomocą zdefiniowanego automatu $\mathfrak{A}$ (funkcja przejścia dla ciągu symboli została zdefiniowana w Dodatku 3).

$$\delta(S, 000023444456\$) = \delta(\delta(S,0), 0002344456\$) = \delta(S_1, 0002344456\$) =$$
$$= \delta(\delta(S_1,0), 002344456\$) = \delta(S_2, 002344456\$) = \delta(\delta(S_2,0), 02344456\$) =$$
$$= \delta(S_3, 02344456\$) = \delta(\delta(S_3,0), 2344456\$) = \delta(S_4, 2344456\$) =$$
$$= \delta(\delta(S_4,2), 344456\$) = \delta(S_5, 344456\$) = \delta(\delta(S_5,3), 44456\$) =$$
$$= \delta(S_6, 44456\$) = \delta(\delta(S_6,4), 4456\$) = \delta(S_7, 4456\$) = \delta(\delta(S_7,4), 456\$) =$$
$$= \delta(S_{12}, 456\$) = \delta(\delta(S_{12},4), 56\$) = \delta(S_{13}, 56\$) = \delta(\delta(S_{13},5), 6\$) =$$
$$= \delta(S_{14}, 6\$) = \delta(\delta(S_{14},6), \$) = \delta(S_{15}, \$) = D.$$

Zakończymy ten podrozdział przedstawieniem głównego algorytmu tej metody. Algorytm zapiszemy w konwencji pascalopodobnej. Wprowadźmy w tym celu następujące elementy:

rec – rozpoznawany obraz (lub *err*, gdy brak decyzji),

state – stan, w którym znajduje się automat,

finalstates – zmienna złożona typu zbiorowego (set), w której zapamiętane są nazwy stanów końcowych.

getchar(ch) – procedura pobiera z bufora wejściowego i podstawia pod zmienną *ch* kolejny znak kodu Freemana,

transfunc(state, ch) – funkcja, która na podstawie aktualnego stanu i pobranego znaku daje nazwę kolejnego stanu

```
procedure FreemRec(var rec);
begin
      state := 'S';
```

```
        repeat
          getchar(ch);
          state := transfunc(state, ch)
        until state in finalstates
        rec := state;
    end
```

10.3. Języki opisu obrazów (PDL) Shawa

Regularne języki oparte na kodach łańcuchowych mają dwie istotne zalety. Po pierwsze obraz $d \in D$ jest opisywany w prosty sposób, jak również prosta jest procedura czytania obrazu $B : D \rightarrow X$ i automatycznego tworzenia kodu $\underline{x} \in X$ (można sobie wyobrażać, że głowica czytająca porusza się wzdłuż pewnego szkieletu lub konturu i na podstawie kierunku ruchu wydaje kolejny symbol x_j kodu $\underline{x}$). Po drugie procedura rozpoznająca $C \cdot F : X \rightarrow I$ (automat deterministyczny $\mathfrak{A}$) jest najprostszą ze znanych tak w sensie jej tworzenia, jak i działania.

Z drugiej strony w trakcie konstrukcji gramatyki daje się zauważyć pewna rozrzutność w ilości produkcji $\mathfrak{p} \in \mathfrak{P}$ (co Czytelnik mógł zobaczyć w szczególnie jaskrawej formie, gdyż ciąg uczący liczył jedynie cztery elementy). Co gorsza, istnieją klasy obrazów D, których wręcz nie można opisać językami regularnymi, ze względu na słabą moc generacyjną gramatyk regularnych. Dlatego też, w tym punkcie przedstawimy języki o istotnie większej mocy opisowej – PDL Shawa.

W przypadku języków Shawa $\mathfrak{L}_{PDL}$, zbiór składowych pierwotnych X nie jest ustalony. Jedynym założeniem jest to, aby każda składowa pierwotna $x_j \in X$ miała wyszczególnione: głowę i ogon. Wyszczególniamy natomiast zbiór pięciu prostych operacji na składowych pierwotnych (omówimy wersję PDL bez operatora indeksowania składowych). Przyjmijmy dwuelementowy zbiór składowych pierwotnych X jak na rysunku 10.2, oraz oznaczmy przez CAT operację katenacji (sklejenia) dwóch składowych. Możemy zdefiniować operację w następujący sposób:

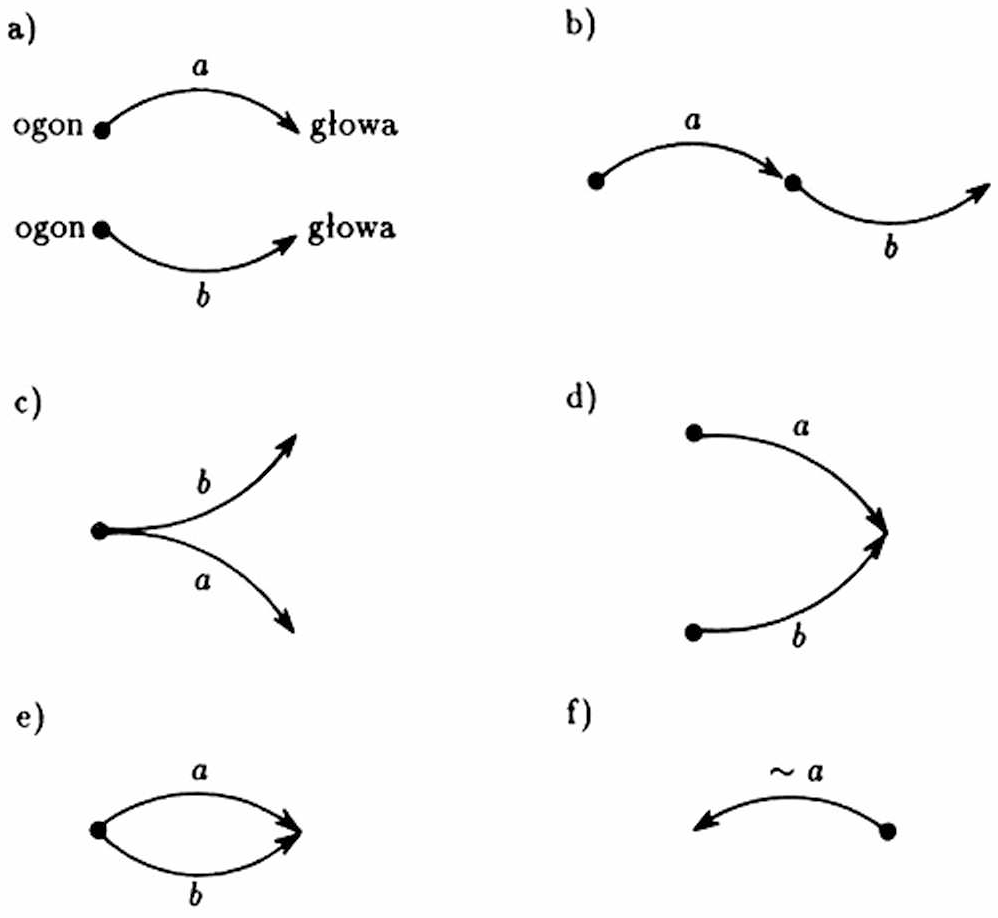
$a + b$ oznacza, że: głowa (a) CAT ogon (b) (rys.10.2b),

$a \times b$ oznacza, że: ogon (a) CAT ogon (b) (rys.10.2c),

$a - b$ oznacza, że: głowa (a) CAT głowa (b) (rys.10.2d),

$a * b$ oznacza, że: ogon (a) CAT ogon (b) oraz głowa (a) CAT głowa (b) (rys.10.2e),

$\sim a$ oznacza odwrócenie zwrotu składowej (rys.10.2f).

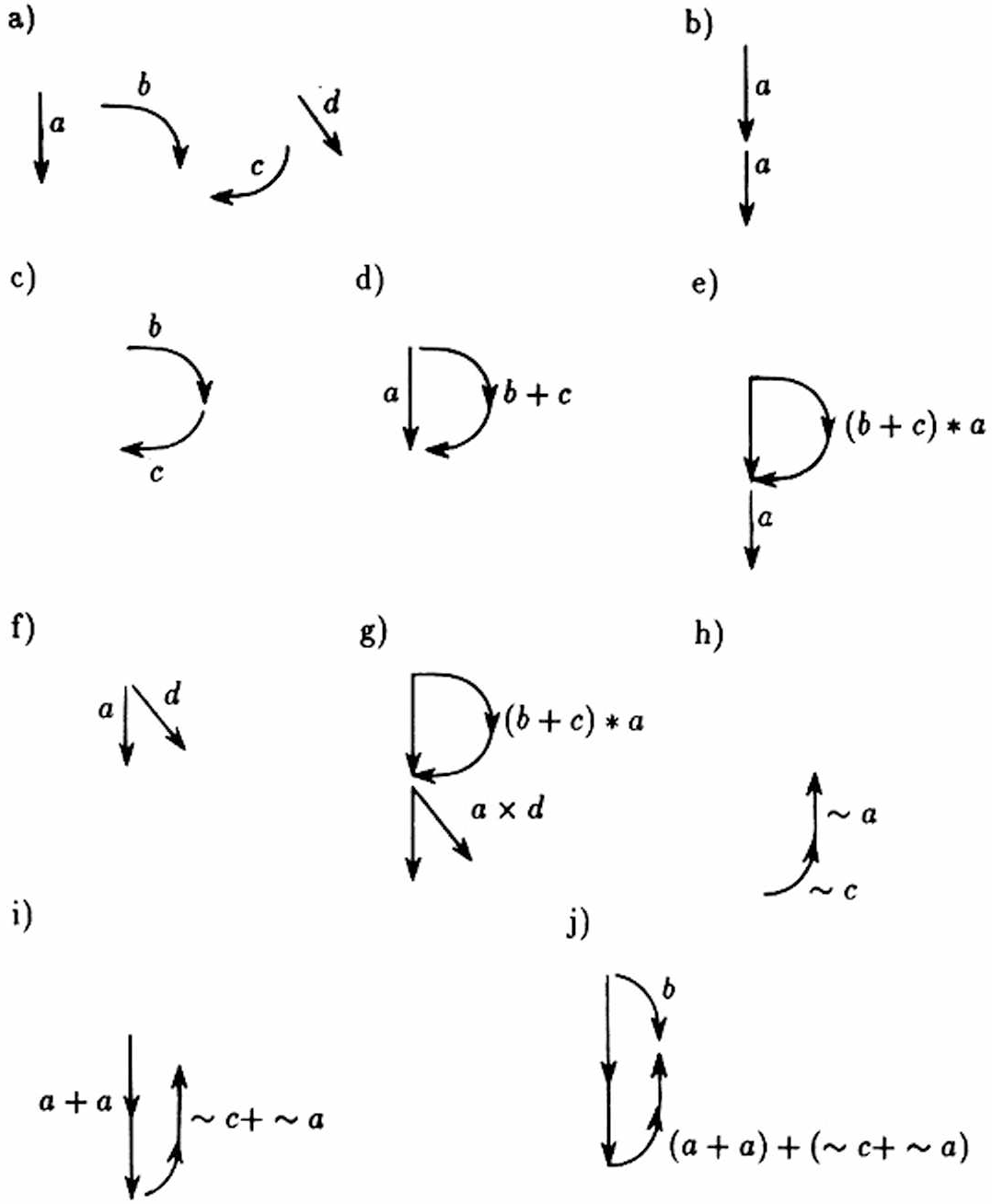


Rys. 10.2. Zbiór składowych pierwotnych oraz różnych form katenacji (sklejania) jako przykład użycia języków Shawa

Metodę Shawa zilustrujemy przykładem rozpoznawania liter rozważanych w poprzednim punkcie (I, P, R, D). W tym celu określimy zbiór składowych pierwotnych (rys. 10.3a). Konstrukcje PDL-reprezentacji przedstawiono na rysunkach 10.3 b-j. Jak widać otrzymujemy następujące wzorce (zaznaczając przez \$ – koniec wzorca):

a) I (b): $a + a\$$,

b) P (c–d–e): $((b + c) * a) + a\$$,



Rys. 10.3. Zbiór składowych pierwotnych i konstrukcje PDL-reprezentacji

c) R (c–d–f–g): $((b + c) * a) + (a \times d)\$$,

d) D (b–h–i–j): $b * ((a + a) + (\sim c + \sim a))\$$.

Bezkontekstową gramatykę $\mathfrak{G}_b$ podklasy $LL(1)$ generującą podane wzorce, konstruujemy w następujący sposób:

$$\mathfrak{G}_b = (\Sigma_N, \Sigma_T, \mathfrak{P}, S),$$

gdzie:

produkcje generujące I:

(1) $S \rightarrow a+a$ (w przypadku gramatyki LL(1) w produkcjach nie uwzględniamy znacznika końca \$),

produkcje generujące P:

(2) $S \rightarrow S_1 + S_2$,

(3) $S_1 \rightarrow (S_3 * a)$,

(4) $S_2 \rightarrow a$,

(5) $S_3 \rightarrow (b + c)$,

produkcje generujące R: (2), (3), (5) oraz

(6) $S_2 \rightarrow (a \times d)$,

produkcje generujące D:

(7) $S \rightarrow S_4 * S_5$,

(8) $S_4 \rightarrow b$,

(9) $S_5 \rightarrow (S_6 + S_7)$,

(10) $S_6 \rightarrow (a + a)$,

(11) $S_7 \rightarrow (\sim c + \sim a)$,

$\Sigma_N = \{S, S_i\}$, $i = 1, 2, \ldots, 7$,

$\Sigma_T = \{a, b, c, d, (,), +, \times, *, \sim\}$.

Teraz, możemy zdefiniować automat $\mathfrak{A}_s$ klasy $LL(1)$ według następujących reguł:

$$\mathfrak{A}_s = (\Sigma_T', \Sigma', \delta, Z_0),$$

gdzie:

(1) $\Sigma_T' := \Sigma_T$,

(2) $\Sigma' := \Sigma_T \cup \Sigma_N \cup \{Z_0\}$,

(3) dla każdego nieterminala $A_1 \in \Sigma_N$ i terminala $a \in \Sigma_T$ takiego, że $a \in first(A_1)$ (patrz Dodatek 3) określamy

$$\delta(A_1, a) := (\sigma, i),$$

gdzie i jest numerem pierwszej produkcji z ciągu produkcji prowadzących do wygenerowania terminala a, natomiast σ jest prawą stroną produkcji,

(4) dla każdego symbolu terminalnego $a \in \Sigma_T$ określamy

$$\delta(a, a) := rem,$$

(5) określamy $\delta(Z_0, \$) := acc$,

(6) dla pozostałych par postaci (A_1, a), które nie zostały rozpatrzone w punktach (3), (4), (5):

$$\delta(A_1, a) := err,$$

gdzie err jest stanem nierozpoznania obrazu.

W przypadku naszej gramatyki mamy:

(1) $\Sigma_T' = \{a, b, c, d, (,), +, \times, *, \sim\}$,

(2) $\Sigma' = \{a, b, c, d, S, S_i, Z_0\}$, $i = 1, \ldots, 7$,

(3) Określamy zbiory typu $first$ dla nieterminali:

$first(S) = \{a, (, b\}$, $first(S_4) = \{b\}$,
$first(S_1) = \{(\}$, $first(S_5) = \{(\}$,
$first(S_2) = \{a, (\}$, $first(S_6) = \{(\}$,
$first(S_3) = \{(\}$, $first(S_7) = \{(\}$

oraz funkcję przejścia:

$\delta(S, a) = (a + a, 1)$, $\delta(S, (\ ) = (S_1 + S_2, 2)$.

(Produkcja (2) jest pierwszą z ciągu prowadzącego do wygenerowania "(": $S - 2 \rightarrow S_1 + S_2 - 3 \rightarrow (S_3 * a) + S_2 \rightarrow \ldots$)

$\delta(S, b) = (S_4 * S_5, 7)$,

$\delta(S_1, (\ ) = ((S_3 * a), 3)$,

$\delta(S_2, a) = (a, 4)$,

$\delta(S_2, (\ ) = ((a \times d), 6)$,

$\delta(S_3, (\ ) = ((b + c), 5),$

$\delta(S_4, b) = (b, 8),$

$\delta(S_5, (\ ) = ((S_6 + S_7), 9),$

$\delta(S_6, (\ ) = ((a + a), 10),$

$\delta(S_7, (\ ) = ((\sim c + \sim a), 11),$

$\delta(a, a) = rem,\ \delta(b, b) = rem,\ \delta(c, c) = rem, \ldots, \delta(\sim, \sim) = rem,$

$\delta(Z_0, \$) = acc,$

natomiast w pozostałych przypadkach funkcja δ przyjmuje wartość *err*.

Rozpoznania odpowiedniego obrazu $d \in D$ (w przypadku jego akceptacji) dokonujemy na podstawie ciągu produkcji $\mathfrak{p}_\mu \in \mathfrak{P}$ jakich użyliśmy do generacji obrazu (a nie stanu końcowego automatu $\mathfrak{A}$, jak to miało miejsce w metodzie kodów Freemana). Ciąg ten jest wypisywany na wyjście automatu. Prześledźmy rozpoznanie litery R poprzez odpowiednie konfiguracje automatu (patrz Dodatek 3).

$$
\begin{array}{rrrl}
 & (\ ((b+c)*a)+(a\times d)\$, & SZ_0,\ \lambda) & \vdash \\
\vdash & (\ ((b+c)*a)+(a\times d)\$, & S_1+S_2Z_0, 2) & \vdash \\
\vdash & (\ ((b+c)*a)+(a\times d)\$, & (S_3*a)+S_2Z_0, 23) & \vdash \\
\vdash & (\ \ (b+c)*a)+(a\times d)\$, & S_3*a)+S_2Z_0, 23) & \vdash \\
\vdash & (\ \ (b+c)*a)+(a\times d)\$, & (b+c)*a)+S_2Z_0, 235) & \vdash \\
\vdash & (\ \ \ b+c)*a)+(a\times d)\$, & b+c)*a)+S_2Z_0, 235) & \vdash \\
\vdash & (\ \ \ \ +c)*a)+(a\times d)\$, & +c)*a)+S_2Z_0, 235) & \vdash \cdots \vdash \\
\vdash & (\ \ \ \ \ \ \ \ (a\times d)\$, & S_2Z_0, 235) & \vdash \\
\vdash & (\ \ \ \ \ \ \ \ (a\times d)\$, & (a\times d)Z_0, 2356) & \vdash \\
\vdash & (\ \ \ \ \ \ \ \ a\times d)\$, & a\times d)Z_0, 2356) & \vdash \cdots \vdash \\
\vdash & (\ \ \ \ \ \ \ \ \ \ \ \ \$, & Z_0, 2356) & \vdash\ acc
\end{array}
$$

Jak widać, litera została zaakceptowana (*acc*) i rozpoznana prawidłowo (ciąg produkcji 2356).

Przedstawimy jeszcze główny algorytm rozpoznający, wprowadzając wcześniej następujące oznaczenia:

rec – rozpoznany obraz (lub *err*, gdy brak decyzji),

tab – tablica, w której zapamiętane są pary postaci (lista numerów produkcji użytych do generacji obrazu *obr*, nazwa obrazu *obr*),

list – lista tworzona w czasie rozpoznawania, w której pamiętane są kolejne numery produkcji podane przez funkcję przejścia,

stack – stos,

initstack – procedura umieszcza na stosie *stack* symbole Z_0 i S,

takein(in) – procedura podstawia pod zmienną *in* wartość pierwszego znaku z bufora wejściowego,

taketop(top) – procedura usuwa ze stosu *stack* i podstawia pod zmienną *top* element znajdujący się na szczycie stosu,

removein – procedura usuwa pierwszy znak z bufora wejściowego,

transproc(*top*, *in*, *upstack*, *out*) – realizacja funkcji przejścia, tzn. procedura na bazie parametrów wejściowych *top* i *in* podstawia pod parametry wyjściowe: *upstack* – prawą stronę produkcji (do umieszczenia jej na stosie), *out* – numer tej produkcji (znakowo); jeśli $top = 'Z_0'$ i $in = 'S'$ to $out := 'a'$; w przypadku błędu pod *out* podstawiany jest znak $'e'$,

push(*upstack*) – procedura umieszcza elementy napisu *upstack* na stosie *stack*,

remember(*out*) – procedura dołącza do listy *list* element *out*,

removelist – procedura odłącza od listy *list* ostatni element,

decide(*list*, *tab*) – funkcja przeszukuje pierwszą kolumnę tablicy *tab* i po znalezieniu napisu równego napisowi zapamiętanemu w liście *list*, daje nazwę obrazu.

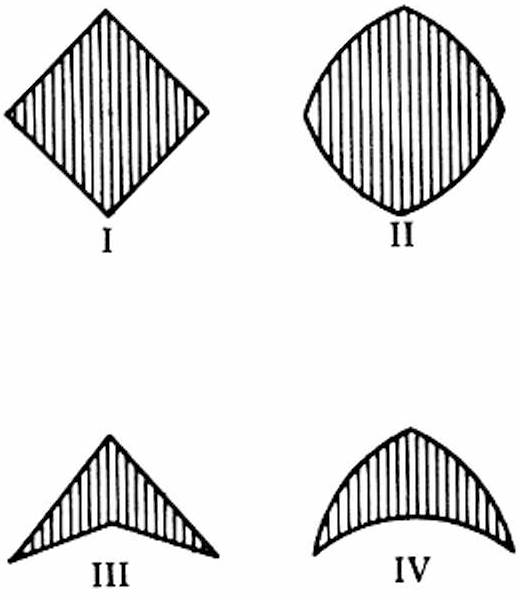
```
procedure ShawRec(var rec);
begin
      initstack;
      repeat
        takein(in);
        taketop(top);
        if top = in then removein
        else
          begin
            transproc(top,in,upstack,out);
            push(upstack);
            remember(out);
          end;
```

```
        until (out = 'a') or (out = 'e');
        if out = 'a' then
          begin
            removelist; {musimy usunąć z listy
                         numerów produkcji ostatni element - 'a'}
            rec := decide(list,tab)
          end
        else rec = 'err';
end.
```

Mechanizm rozpoznający przedstawiliśmy dla klasy $LL(1)$ gramatyk bezkontekstowych. Może się zdarzyć, że pewne klasy obrazów $d \in D$ opisywanych przez języki Shawa nie dadzą się wygenerować przez gramatyki $LL(1)$. Wtedy, rozwiązaniem tego problemu jest konstrukcja mniej restryktywnej gramatyki, a więc $LL(k)$, $k > 1$ [22,23], $LR(k)$ [27], czy też precedensyjnej [28] oraz automatu odpowiedniego typu.

10.4. Języki opisu cech kształtów (Jakubowski)

Przedstawione w poprzednich punktach metody pozwalają na rozpoznanie kształtów pewnych obiektów d i określenie ich przynależności do ustalonych obrazów $D^i \subseteq D$. Nie dają jednak możliwości oparcia procedury rozpoznającej $\hat{A}$ na analizie cech $\underline{x} \in X$ określających istotę tych kształtów (zagłębienie w pewnym miejscu obiektu, wypukłość czy wklęsłość fragmentu konturu ograniczającego obiekt itd). Możliwość taką dają, natomiast, języki opisu cech kształtów $\mathfrak{L}_{SFDL}$. Na bazie opisu SFDL możliwa jest również automatyczna segmentacja złożonych obiektów na obszary quasi-wypukłe, co z kolei pozwala utworzyć syntaktyczne drzewo opisu obiektu. Analiza takich drzew jest narzędziem do porównywania obiektów w pewnych ich podobszarach. Obraz na poziomie składowych pierwotnych jest rozpoznawany przez deterministyczny *transdjuser* o skończonej liczbie stanów, który definiuje opis cech kształtu na poziomie deskryptorów. Rozpoznanie cech kształtu x_j opisanych przez deskryptory może być dokonywane przez deterministyczny automat $\mathfrak{A}_d$ o skończonej ilości stanów. Istnieje wiele rozszerzeń tej metody takich, jak np. języki atrybutowane, języki niewrażliwe na pewne transformacje obrazu (np. obroty), analiza syntaktyczna z korekcją błędów, samouczący analizator syntaktyczny [39]. W niniejszym opracowaniu przedstawimy tylko dwa najniższe poziomy (analizy na poziomie



Rys. 10.4. Zbiór obiektów podlegających opisowi w języku Jakubowskiego

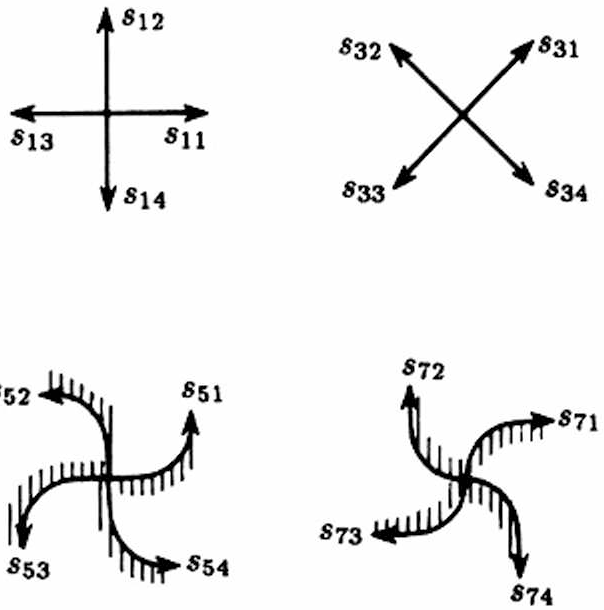
składowych pierwotnych i rozpoznawania na poziomie deskryptorów – bez wyodrębniania rejonów, zagłębień, itp.).

Rozważmy przykładowo, cztery obiekty przedstawione na rysunku 10.4. Jak łatwo zauważyć, dwa górne obiekty można uważać za należące do tej samej klasy, a dwa dolne do innej. Ponieważ proces rozpoznawania tych obiektów prześledzimy na dwóch najniższych poziomach abstrakcji metody Jakubowskiego, więc wprowadzimy pewne uproszczenia w przytaczanych formalizmach (np. nie będzie rozróżnienia pomiędzy wklęsłym i wypukłym *biquadem*).

Zbiór składowych pierwotnych typu *short* przedstawiono na rysunku 10.5, w kolejności – segmenty: osiowe (s_{1j}), pochyłe (s_{3j}), wklęsłe (s_{5j}) i wypukłe (s_{7j}), $j = 1,2,3,4$. Składowe pierwotne typu *long* (które są, jak łatwo można się domyśleć, dłuższe) mają kody (odpowiednio): s_{2j}, s_{4j}, s_{6j}, s_{8j}, $j = 1,2,3,4$. Za pomocą tak zdefiniowanych składowych, możemy zakodować rozważane obrazy w następujący sposób (punkt startowy został zaznaczony kropką na rysunku 10.6):

$$\text{Des(I)} = s_{31}s_{31}s_{31}s_{34}s_{34}s_{34}s_{33}s_{33}s_{33}s_{32}s_{32}s_{32},$$

$$\text{Des(II)} = s_{71}s_{31}s_{71}s_{74}s_{34}s_{74}s_{73}s_{33}s_{73}s_{72}s_{32}s_{72},$$



Rys. 10.5. Zbiór składowych pierwotnych typu *short*

$$\text{Des(III)} = s_{31}s_{31}s_{31}s_{34}s_{34}s_{34}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},$$

$$\text{Des(IV)} = s_{71}s_{31}s_{71}s_{74}s_{34}s_{74}s_{52}s_{32}s_{52}s_{53}s_{33}s_{53}.$$

Mając tak opisane obrazy moglibyśmy skonstruować gramatykę regularną i automat, jak to miało miejsce dla metody Freemana. W przypadku omawianej metody dokonamy wcześniej przetwarzania opisu konturu w celu wyodrębnienia pewnych cech tego konturu.

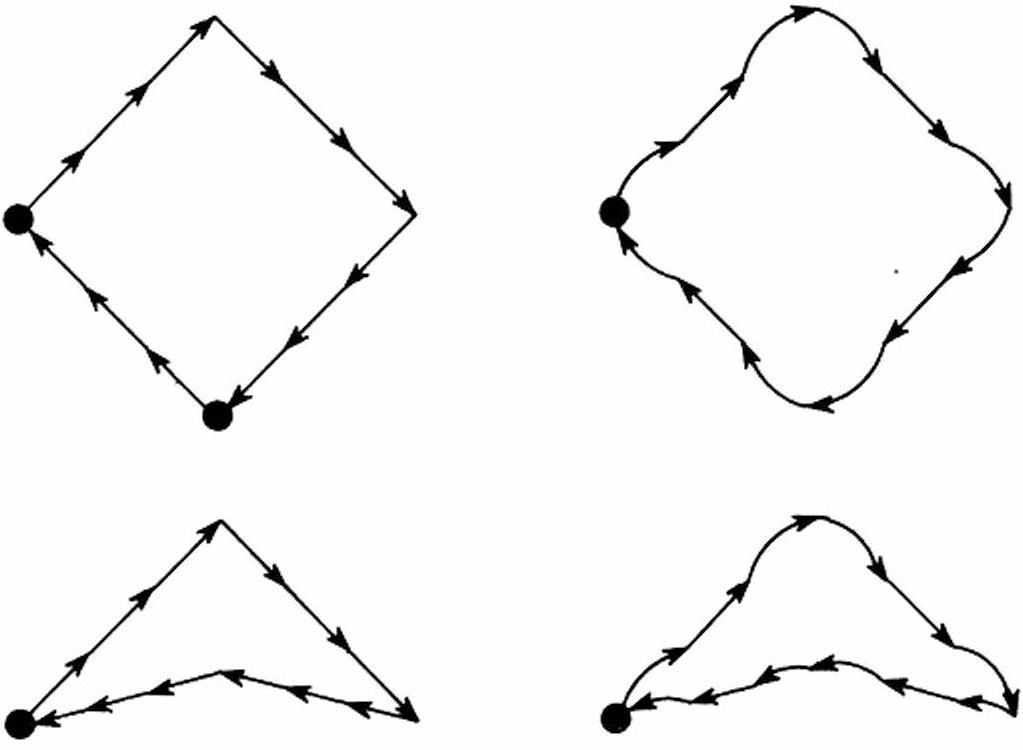
Fragment konturu opisany kodem składającym się z elementów s_{ij}, gdzie $i = 1, 2, \ldots, 8$, a j jest ustalone nazywamy (j)-sinquadem. Przykładowo, fragment obrazu IV z rysunku 10.6 pomiędzy punktami A i B tworzy (2)-sinquad. Przetwarzanie, o którym mowa, polega na utworzeniu nowego opisu, w którym będą zaznaczone jedynie przejścia pomiędzy sinquadami. Użyjemy do tego celu sekwencyjnego transdjusera z wielokrotnym wejściem (patrz dodatek 3) zdefiniowanego w następujący sposób:

$$ST = (Q, \Sigma_T, \Delta, \delta, Q_0),$$

gdzie:

$$Q = \{q_1, q_2, q_3, q_4\},$$

$$\Sigma_T = \{s_{ij}, \quad i = 1, \ldots, 8, \quad j = 1, \ldots, 4\},$$



Rys. 10.6. Opis obiektów z rys. 10.4 za pomocą składowych z rys. 10.5

$\Delta = \{1, 2, 3, 4\}$,

$Q_0 = \{q_1, q_2, q_3, q_4\}$,

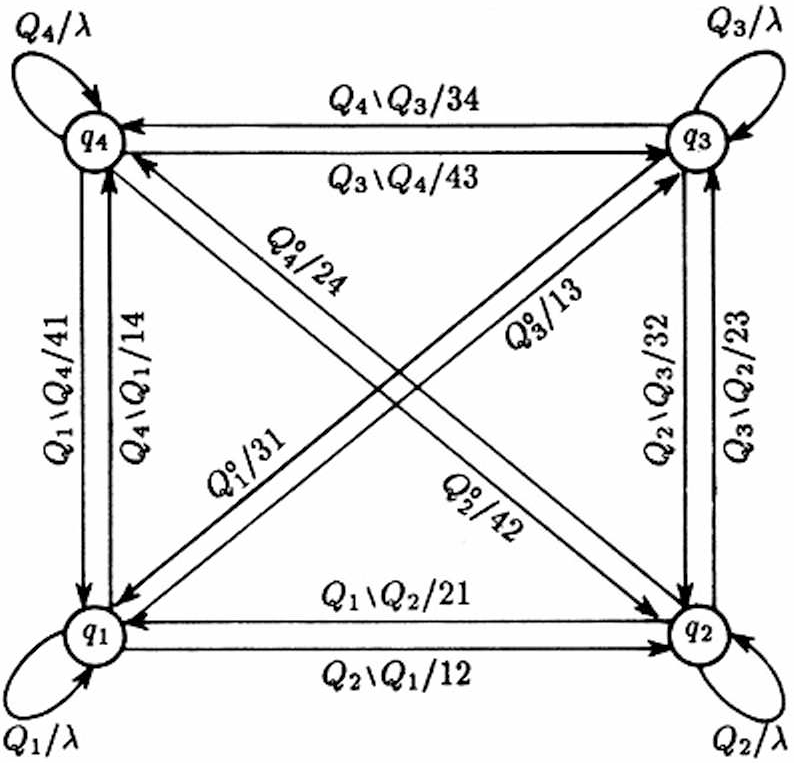
a δ jest zdefiniowana przez rysunek 10.7.

Krawędź diagramu (rys. 10.7) łącząca stan q_i ze stanem q_j oraz etykietowana przez $Q_j \backslash Q_i / ij$ oznacza, że przejście ze stanu q_i do stanu q_j następuje wtedy, gdy wczytana została składowa należąca do (j)-sinquadu, a nie należąca do (i)-sinquadu (składowe osiowe należą do dwóch sąsiednich sinquadów równocześnie). Wówczas też na wyjściu wypisywany jest napis ij (przejście z (i)-sinquadu do (j)-sinquadu).

Krawędź etykietowana przez Q_i/λ oznacza, że pojawiła się składowa należąca do (i)-sinquadu (nic nie wypisujemy na wyjście).

Krawędź etykietowana przez Q_j°/ij oznacza, że pojawiła się składowa należąca do (j)-sinquadu nie będąca składową osiową.

Jak łatwo zauważyć, wczytując opisy naszych obiektów otrzymamy na wyjściu:



Rys. 10.7. Graf stanów transdjusera

ST(Des(I)) = 14.43.32.21,

ST(Des(II)) = 14.43.32.21,

ST(Des(III)) = 14.42.23.31,

ST(Des(IV)) = 14.42.23.31.

Zatem obrazy I oraz II należą do tej samej klasy, a obrazy III oraz IV do innej.

Prawostronnie regularna gramatyka generująca obie klasy ma postać:

$$\mathfrak{G}_{pr} = (\Sigma_N, \Sigma_T, \mathfrak{P}, S),$$

gdzie produkcje $\mathfrak{P}$ generujące obrazy I oraz II (każdy napis zakończymy przez \$):

(1) $S \rightarrow 14\ S_1$,
(2) $S_1 \rightarrow 43\ S_2$,
(3) $S_2 \rightarrow 32\ S_3$
(4) $S_3 \rightarrow 21\ S_4$,
(5) $S_4 \rightarrow \$$,

produkcje $\mathfrak{P}$ generujące obrazy III oraz IV: (1) oraz

(6) $S_1 \to 42\, S_5$ (8) $S_6 \to 31\, S_7$,

(7) $S_5 \to 23 S_6$ (9) $S_7 \to \$$,

$\Sigma_N = \{S, S_i\}$, $i = 1, \ldots, 7$,

$\Sigma_T = \{\$, 14, 43, 32, 21, 42, 23, 31\}$.

Natomiast automat $\mathfrak{A}$ jest skonstruowany w następujący sposób:

$\mathfrak{A} = (\Sigma'_T, Q, \delta, q, F)$,

$\Sigma'_T = \{\$, 14, 43, 32, 21, 42, 23, 31\}$,

$Q = \{S, S_i\}$, $i = 1, \ldots, 7$,

$q_0 = S$,

$F = \{A, B, \text{err}\}$,

gdzie klasa A zawiera obrazy I i II, klasa B – obrazy III i IV;

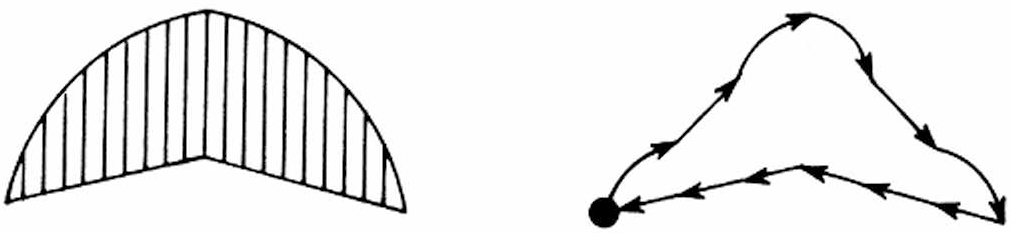
$\delta(S, 14) = S_1$, $\delta(S_1, 42) = S_5$,

$\delta(S_1, 43) = S_2$, $\delta(S_5, 23) = S_6$,

$\delta(S_2, 32) = S_3$, $\delta(S_6, 31) = S_7$,

$\delta(S_3, 21) = S_4$, $\delta(S_7, \$) = B$.

$\delta(S_4, \$) = A$,



Rys. 10.8. Obiekt do rozpoznawania (po lewej) i jego opis (po prawej)

Dokonajmy rozpoznania obrazu V znajdującego się na rysunku 10.8, różnego od wcześniej rozważanych obrazów:

Des(V) = $s_{71}s_{31}s_{71}s_{74}s_{34}s_{74}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33}$.

Zdefiniujmy konfigurację *transdjusera* jako trójkę:

(aktualny stan, wejście, wyjście).

Przetwarzanie Des(V) będzie przebiegać następująco:

$(q_1,\ s_{71}s_{31}s_{71}s_{74}s_{34}s_{74}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},\ \lambda\ ) \vdash$

$\vdash (q_1,\ s_{31}s_{71}s_{74}s_{34}s_{74}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},\ \lambda\ ) \vdash$

$\vdash (q_1,\ s_{71}s_{74}s_{34}s_{74}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},\ \lambda\ ) \vdash$

$\vdash (q_1,\ s_{74}s_{34}s_{74}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},\ \lambda\ ) \vdash$

$\vdash\!\!-$ $(q_4,$ $s_{34}s_{74}s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},$ 14 $)$ $\vdash\!\!-\cdots\vdash\!\!-$
$\vdash\!\!-$ $(q_4,$ $s_{32}s_{32}s_{32}s_{33}s_{33}s_{33},$ 14 $)$ $\vdash\!\!-$
$\vdash\!\!-$ $(q_2,$ $s_{32}s_{32}s_{33}s_{33}s_{33},$ 14.42 $)$ $\vdash\!\!-\cdots\vdash\!\!-$
$\vdash\!\!-$ $(q_2,$ $s_{33}s_{33}s_{33},$ 14.42 $)$ $\vdash\!\!-\cdots\vdash\!\!-$
$\vdash\!\!-$ $(q_3,$ $s_{33}s_{33},$ 14.42.23$)$ $\vdash\!\!-\cdots\vdash\!\!-$
$\vdash\!\!-$ $(q_3,$ $\lambda,$ 14.42.23$)$ $\vdash\!\!-\cdots\vdash\!\!-$

$\vdash\!\!-\ldots$ (ponieważ punkt startowy był granicą pomiędzy (3)-sinquadem a (1)-sinquadem, więc na wyjście wypisujemy jeszcze 31) $\cdots\vdash\!\!-(q_1, \lambda,$ 14.42.23.31)

Zatem: ST(Des(V)) = 14.42.23.31.

Rozpoznajmy ST(Des(V)) za pomocą tak zdefiniowanego automatu:

$$\delta(S, 14.42.23.31\$) = \delta(\delta(S, 14), 42.23.31\$) = \delta(S_1, 42.23.31\$) =$$

$$= \delta(\delta(S_1, 42), 23.31\$) = \delta(S_5, 23.31\$) = \delta(\delta(S_5, 23), 31\$) =$$

$$= \delta(S_6, 31\$) = \delta(\delta(S_6, 31), \$) = \delta(S_7, \$) = B.$$

Jak widać, obraz został zaklasyfikowany do klasy B, co jest zgodne z naszymi intuicjami.

Przedstawmy jeszcze algorytm *transdjusera* przetwarzającego początkowy opis konturu (algorytm automatu deterministycznego, którego użyliśmy do rozpoznawania został opisany w p. 10.1), wprowadzając

bufin – bufor wejściowy,

actsinq – numer bieżącego sinquadu,

newsinq – numer nowego sinquadu,

firstsinq – numer sinquadu, do którego należał pierwszy analizowany element,

givesinquad(bufin) – funkcja pobiera z bufora kolejny element i daje numer sinquadu, do którego ten element należy,

putbufout($\text{sinq}_1, \text{sinq}_2$) – procedura na podstawie numerów sinquadów $\text{sinq}_1, \text{sinq}_2$ zapisuje do bufora wyjściowego przejście pomiędzy nimi w postaci $\text{sinq}_1\text{sinq}_2$,

eof(buf) – funkcja przyjmuje wartość **true**, kiedy bufor *buf* jest pusty.

```
procedure RecJakubowski;
begin
      actsinq := givesinquad(bufin);
      firstsinq := actsinq;
      repeat
      newsinq := givesinq(bufin);
      if actsinq ≠ newsinq then
      begin
        putbufout(actsinq,newsinq);
        actsinq := newsinq
      end;
      until eof(bufin);
      if newsinq ≠ firstsinq then putbufout(newsinq,firstsinq);
      { dołożenie przejścia pomiędzy sinquadami w przypadku
      gdy analiza startowała z punktu granicznego pomiędzy sinquadami }
end.
```

11. METODY DRZEWOWE

Metody drzewowe rozpoznawania obrazów są konstruowane na bazie teorii języków drzewowych i automatów drzewowych. Teoria ta, będąca działem lingwistyki matematycznej, doczekała się wielu publikacji, a nawet opracowań monograficznych (przegląd tych prac Czytelnik może znaleźć w [19]). Automaty rozważane w teorii automatów drzewowych są bardziej skomplikowane niż automaty ciągowe (co jest rzeczą naturalną, zważywszy, że drzewo jest bardziej złożoną strukturą niż ciąg). W zastosowaniach do rozpoznawania obrazów używany jest najprostszy typ automatu drzewowego – deterministyczny automat czytający drzewo od brzegu (liści) do korzenia (ang. *deterministic frontier-to-root tree recognizer*), czyli automat $\mathfrak{A}_{DF}$ [30], który określany jest dla ekspansywnej gramatyki drzewowej [29].

W rozdziale przedstawimy dwie metody rozpoznawania oparte na wspomnianym automacie $\mathfrak{A}_d$. Pierwsza z nich jest metodą analizy syntaktycznej drzew o zaetykietowanych i skierowanych krawędziach, czyli drzew EDT (ang. *Edge-labelled Directed Tree)*. Druga – to metoda analizy syntaktycznej drzew prostych, tzn. takich, których krawędzie nie są zaetykietowane ani skierowane, czyli drzew T. Metoda pierwsza może być wykorzystana do analizy sceny [32], druga – do analizy faktur [33] (oba problemy zostały zdefiniowane w rozdziale 9)([1]).

11.1. Analiza syntaktyczna drzew EDT

Rozważmy trzy obrazy (sceny) znajdujące się na rysunku 11.1. Przyjmując zbiory składowych pierwotnych reprezentujących: obiekty scen i relacje pomiędzy obiektami jak na rysunku 11.1a, możemy utworzyć reprezentacje

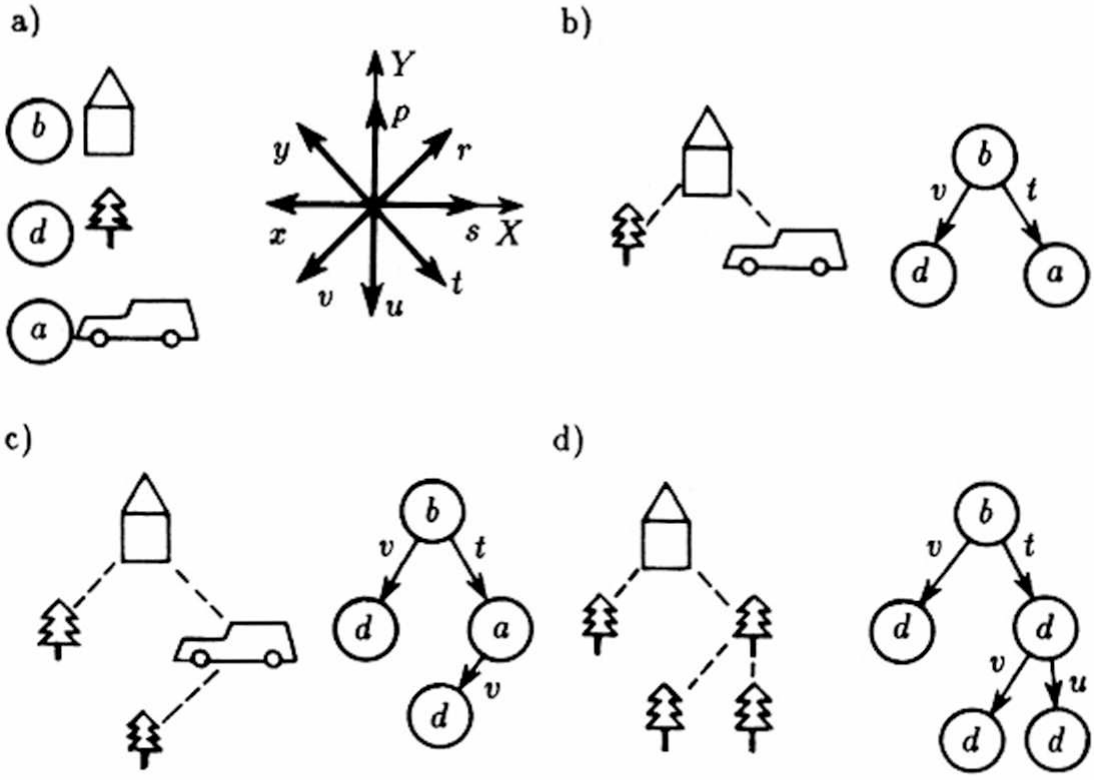
([1]) Definicje formalne: ekspansywnych gramatyk drzewowych i automatów DF znajdują się w Dodatku 4.

drzewowe widoczne obok wspomnianych scen. Zapisując te trzy drzewa-reprezentacje w postaci nawiasowej (patrz rozdział 9), otrzymamy:

I scena – $b(vd\ ta)$,

II scena – $b(vd\ ta(vd))$,

III scena – $b(vd\ td(vd\ ud))$.



Rys. 11.1. Składowe pierwotne opisu drzewowego oraz trzy sceny opisane strukturalnie jako drzewa (opis w tekście)

Zdefiniujmy teraz gramatykę $\mathfrak{G}_{EDT}$ generującą rozważane sceny, na bazie której będziemy mogli skonstruować automat $\mathfrak{A}_d$ rozpoznający te sceny. Gramatyka ta jest piątką (patrz Dodatek 4):

$$\mathfrak{G}_{EDT} = (\Sigma, r, \Gamma, \mathfrak{P}, Z),$$

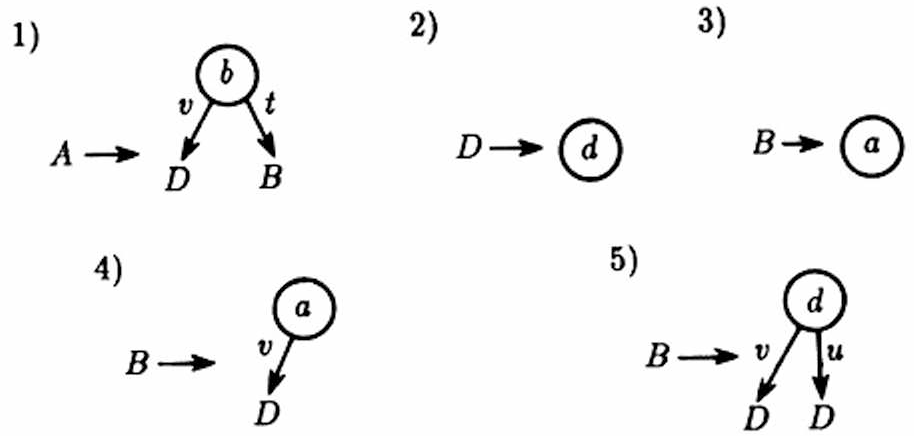
gdzie zbiór produkcji $\mathfrak{P}$ przedstawiony na rysunku 11.2 jest określony w następujący sposób:

(1) $A \rightarrow b(vDtB)$, (2) $D \rightarrow d$,

(3) $B \to a$, (4) $B \to a(vD)$,
(5) $B \to d(vDuD)$,

zaś pozostałe elementy zdefiniowano następująco:

$\Sigma = \Sigma_T \cup \Sigma_N$, $\Sigma_T = \{b, d, a\}$, $\Sigma_N = \{A, B, D\}$,
$\Gamma = \{v, t, u\}$, $Z = \{A\}$.



Rys. 11.2. Zbiór produkcji gramatyki EDTG

Przykładowo, generacja drugiej rozważanej sceny (z rysunku 11.1c) zostanie zapisana, następująco (cyfry w strzałkach oznaczają numery stosowanych produkcji):

$A - (1) \to b(vD\,tB) - (2) \to b(vdtB) - (4) \to b(vd\,ta(vD)) - (2) \to$
$- (2) \to b(vd\,ta(vd))$.

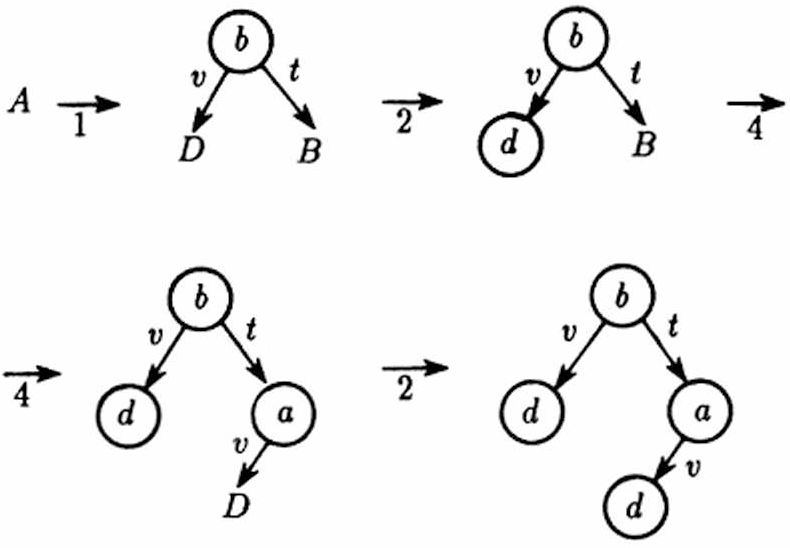
Ilustruje to rysunek 11.3.

Natomiast automat $\mathfrak{A}_{DFEDT}$ nad zbiorem etykiet wierzchołkowych $\Sigma_T = \{a_1, \ldots, a_n\}$ i zbiorem etykiet krawędziowych (p. Dodatek 4) rozpoznający rozważane sceny, określimy dla gramatyki $\mathfrak{G}_{EDT} = (\Sigma, r, \Gamma, \mathfrak{P}, Z)$ według następujących reguł:

$$\mathfrak{A}_{DFEDT} = (Q, \delta_1, \ldots, \delta_n, F),$$

gdzie:

(1) $Q := \Sigma - \Sigma_T$,



Rys. 11.3. Generacja sceny zgodnie z zasadami gramatyki drzewowej

(2) $F := Z$,

(3) dla każdej produkcji postaci $A \to a(\tau_1 A_1 \ldots \tau_{r(a)} A_{r(a)}) \in \mathfrak{P}$ o numerze i, funkcję przejścia δ_a na $\tau_1 A_1, \ldots, \tau_{r(a)} A_{r(a)}$ definiujemy w następujący sposób:

$$\delta_a(\tau_1 A_1, \ldots, \tau_{r(a)} A_{r(a)}) = (A, i) \qquad (\text{dla} \quad A \to a \in \mathfrak{P} : \delta_a = (A, i)).$$

W naszym przypadku mamy:

(1) $Q = \{A, B, D\}$,

(2) $F = \{A\}$,

(3) $\delta_b(vD, tB) = (A, 1)$, $\quad \delta_d = (D, 2)$, $\quad \delta_a = (B, 3)$,
$\delta_a(vD) = (B, 4)$, $\quad \delta_d(vD, uD) = (B, 5)$.

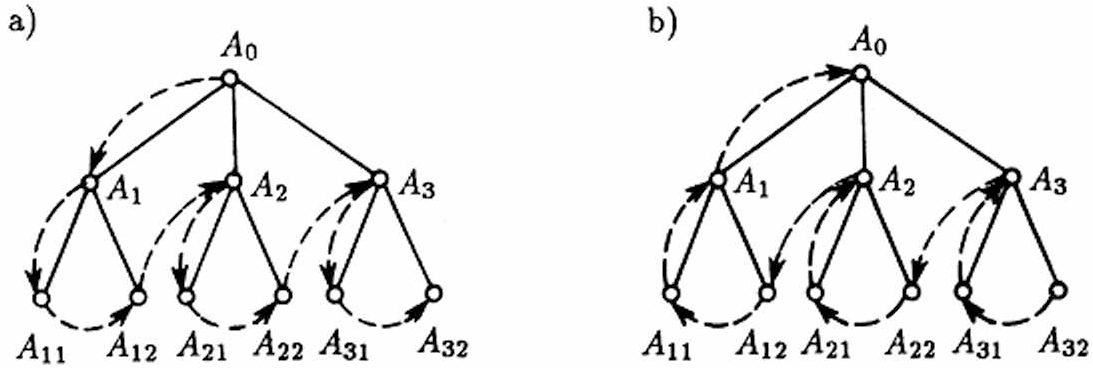
Prześledźmy rozpoznawanie wskazanej sceny (z rys 11.1 c), obserwując przejścia pomiędzy konfiguracjami automatu $\mathfrak{A}_{DFEDT}$ (patrz Dodatek 4).

$$(rp(b(vd\,ta(vd))), \lambda) \vdash (\delta_b(v\,rp(d), t\,rp(a(vd))), \lambda) \vdash$$
$$\vdash (\delta_b(v\delta_d, t\delta_a(rp(vd))), \lambda) \vdash (\delta_b(v\delta_d, t\delta_a(v\,rp(d))), \lambda) \vdash$$
$$\vdash (\delta_b(v\delta_d, t\delta_a(v\delta_d)), \lambda) \vdash (\delta_b(v\delta_d, t\delta_a(vD)), 2) \vdash$$
$$\vdash (\delta_b(v\delta_d, tB), 24) \vdash (\delta_b(vD, tB), 242) \vdash (A, 2421).$$

Jak widać, scena została zaakceptowana ($A \in F$) i rozpoznana prawidłowo (ciąg produkcji: 1242 – czytany od prawej do lewej).

Zwróćmy jeszcze uwagę na dwa ograniczenia, które do tej pory zakładaliśmy milcząco. Pierwsze dotyczy zbioru produkcji $\mathfrak{P}$ gramatyki $\mathfrak{G}_{EDT}$. Otóż dla każdych dwóch produkcji $\mathfrak{p}_1$ i $\mathfrak{p}_2$ mających taką samą prawą stronę $\tau : A_1 \rightarrow \tau$, $A_2 \rightarrow \tau$, musi zachodzić warunek: $A_1 \neq A_2$. W przeciwnym razie proces analizy w automacie $\mathfrak{A}_{DFEDT}$ miałby charakter niedeterministyczny.

Drugie ograniczenie dotyczy sposobu generowania i analizy drzewa. Zauważmy, że drzewo generowaliśmy w kolejności: korzeń, najbardziej lewe poddrzewo,...,najbardziej prawe poddrzewo (rysunek 11.3). Zasada ta została zilustrowana rysunkiem 11.4a. Najpierw generujemy terminal A_0 i nieterminale A_1, A_2, A_3 ($S \rightarrow A(A_1 A_2 A_3)$). Następnie zajmujemy się generacją poddrzewa rozpoczynającego się w skrajnie lewym wierzchołku A_1 i dopiero po wygenerowaniu całego poddrzewa (w którym rekurencyjnie stosujemy podaną zasadę), zajmujemy się wierzchołkiem A_2 (tzn. poddrzewem rozpoczynającym się w nim), itd. Natomiast analizy dokonujemy w odwrotnej kolejności, tzn. od skrajnie prawego liścia poruszamy się w drzewie z prawa – na lewo i z dołu – do góry, co zostało pokazane na rysunku 11.4b.



Rys. 11.4. Kolejność: a) generacji poddrzew, b) analizy poddrzew w gramatyce drzewowej

Zakończymy rozważania prezentacją algorytmu analizy sceny przedstawioną metodą. Wprowadzimy w nim następujące elementy:

zmienne:

rec, tab, list – opisano w p. 10.2,

finalstates – opisano w p. 10.1,

procedury:

remember(out), **decide**(list,tab) – opisano w p. 10.2,

findnode(place,label) – procedura znajduje kolejny wierzchołek drzewa, który powinien być obecnie analizowany (zgodnie z przedstawioną zasadą analizy) – parametr *place* daje miejsce (adres) tego wierzchołka, a *label* – jego etykietę,

transfunc(label, place, nonterminal, out) – realizacja funkcji przejścia, tzn. procedura na bazie parametrów wejściowych: *label* (wybierającego właściwą funkcję przejścia – δ_{label}) i *place* (wskazującego argument funkcji δ_{label} – czyli $\tau_1 A_1, \ldots, \tau_{r(a)} A_{r(a)}$, gdy $label(\tau_1 A_1 \ldots \tau_{r(a)} A_{r(a)})$ jest poddrzewem rozpoczynającym się w wierzchołku wskazanym przez *place*), daje: symbol do jakiego dokonujemy redukcji - *nonterminal* oraz numer produkcji – *out* (znakowo); w przypadku gdy takie przejście nie istnieje – pod *out* podstawiany jest znak e,

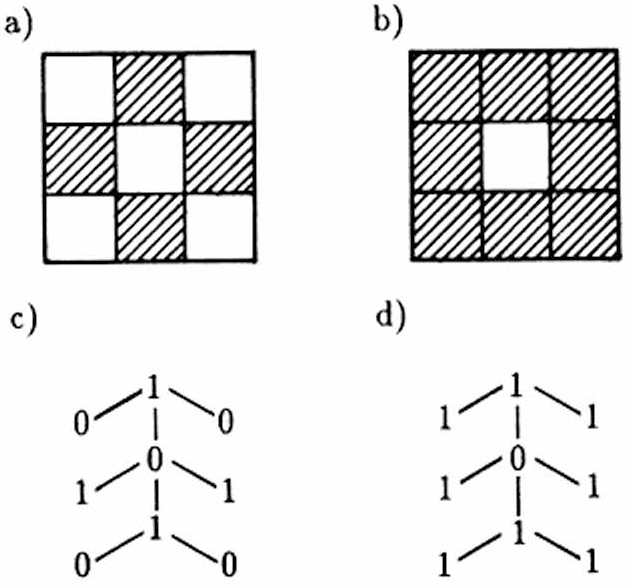
replace(place, nonterminal) – procedura zastępuje napis label ($\tau_1 A_1 \ldots \tau_{r(a)} A_{r(a)}$) wskazany przez parametr place napisem *nonterminal*.

```
procedure TreeRec (var rec);
begin
      repeat
        findnode(place,label);
        transfunc(label,place,nonterminal,out);
        replace(place,nonterminal);
        remember(out);
      until (nonterminal in finalstates) or (out = 'e');
      if out = 'e' then rec := 'err'
                   else rec := decide(list,tab);
end;
```

Algorytm ten został skonstruowany na bazie automatu drzewowego przedstawionego w tym rozdziale i stanowi jego wierne odwzorowanie. Innym rozwiązaniem, mniej zgodnym z tradycją teorii języków formalnych, za to bardziej naturalnym dla struktury drzewowej, byłby algorytm rekurencyjny. W tym celu sekwencję wywoływania procedur transfunc, replace oraz remember należałoby zgrupować w osobnej procedurze, np. **analysis**(label, place, nonterminal, out). Następnie pętlę repeat należałoby zastąpić rekurencyjnym wywołaniem procedury analysis po wcześniejszym ustaleniu jej parametrów wejściowych przez procedurę **findnode**.

11.2. Analiza syntaktyczna drzew T

W rozdziale 9 omówiliśmy problem analizy faktur, ilustrując go rysunkami: pewnej faktury, jej wzorca umieszczonego w okienku oraz drzewa reprezentującego ten wzorzec (rysunki 9.8 – 9.10). W niniejszym rozdziale rozważymy dwa bardzo proste wzorce (I oraz II) faktur, przedstawione na rysunkach 11.5a i b (zapis nawiasowy wzorca z rysunku 9.8b nie zmieściłby się w jednej linii tekstu). Reprezentacje drzewowe tych wzorców zostały utworzone zgodnie z zasadą przedstawioną w rozdziale 9 (rysunek 9.9):



Rys. 11.5. Dwa proste wzorce faktur i ich reprezentacje drzewowe

I wzorzec – 1(00(11(00)1)0) (rys. 11.5c),

II wzorzec – 1(10(11(11)1)1) (rys. 11.5d).

Gramatykę $\mathfrak{G}_T$ generująca oba wzorce zdefiniowano w następujący sposób:

$$\mathfrak{G}_T = (\Sigma, r, \mathfrak{P}, Z),$$

gdzie zbiór $\mathfrak{P}$ składa się z produkcji:

(1) $A \to 1(ZBZ)$, (2) $B \to 0(WCW)$,
(3) $C \to 1(ZZ)$, (4) $Z \to 0$,
(5) $W \to 1$, (6) $A \to 1(WDW)$,
(7) $D \to 0(WEW)$, (8) $E \to 1(WW)$,

a pozostałe elementy są następujące:

$$\Sigma_T = \{1, 0\}, \qquad \Sigma_N = \{A, B, C, D, E, Z, W\}, \qquad Z = \{A\}.$$

Wzorzec I można wygenerować następująco (cyfry w strzałkach oznaczają numery stosowanych produkcji):

$$A \xrightarrow{(1)} 1(ZBZ) \xrightarrow{(2)} 1(0BZ) \xrightarrow{(2)} 1(00(WCW)Z) \xrightarrow{(5)} \xrightarrow{(5)} 1(00(1CW)Z) \xrightarrow{(3)} 1(00(11(ZZ)W)Z) \xrightarrow{(4)} \xrightarrow{(4)} 1(00(11(0Z)W)Z) \xrightarrow{(4)} 1(00(11(00)W)Z) \xrightarrow{(5)} \xrightarrow{(5)} 1(00(11(00)1)Z) \xrightarrow{(4)} 1(00(11(00)1)0)$$

Podamy teraz reguły konstrukcji automatu $\mathfrak{A}_{DFT}$ nad zbiorem etykiet wierzchołkowych $\Sigma_T = \{a_1, \ldots, a_n\}$ rozpoczynającego wzorce generowane przez gramatykę $\mathfrak{G}_T = (\Sigma, r, \mathfrak{P}, Z)$:

$$\mathfrak{A}_{DFT} = (q, \delta_1, \ldots, \delta_n, F),$$

gdzie:

(1) $Q := \Sigma - \Sigma_T$,

(2) $F := Z$,

(3) dla każdej produkcji postaci $A \to a(A_1 \ldots A_{r(a)}) \in \mathfrak{P}$ o numerze i, funkcję przejścia δ_a na $A_1, \ldots, A_{r(a)}$ definiujemy w następujący sposób:

$$\delta(A_1, \ldots, A_{r(a)}) = (a, i) \qquad (\text{dla } A \to a \in P : \delta_a = (A, i)).$$

Skonstruujmy automat $\mathfrak{A}_{DFT}$ rozpoznający rozważane wzorce:

(1) $Q = \{A, B, C, D, E, Z, W\}$,

(2) $F = \{A\}$,

(3) $\delta_1(Z, B, Z) = (A, 1)$, $\quad \delta_1 = (W, 5)$,
$\delta_0(W, C, W) = (B, 2)$, $\quad \delta_1(W, D, W) = (A, 6)$,
$\delta_1(Z, Z) = (C, 3)$, $\quad \delta_0(W, E, W) = (D, 7)$,
$\delta_0 = (Z, 4)$, $\quad \delta_1(W, W) = (E, 8)$.

Rozpoznawanie wzorca I przebiega następująco (patrz Dodatek 4):

$$(\mathrm{rp}(1(00(11(00)1)0)), \lambda) \vdash$$
$$\vdash (\delta_1(\mathrm{rp}(0), \mathrm{rp}(0(11(00)1), \mathrm{rp}(0)), \lambda) \vdash$$
$$\vdash (\delta_1(\delta_0, \delta_0(\mathrm{rp}(1), \mathrm{rp}(1(00)), \mathrm{rp}(1)), \delta_0), \lambda) \vdash$$
$$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, \delta_1(\mathrm{rp}(0), \mathrm{rp}(0)), \delta_1), \delta_0), \lambda) \vdash$$
$$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, \delta_1(\delta_0, \delta_0), \delta_1), \delta_0), \lambda) \vdash$$

$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, \delta_1(\delta_0, \delta_0), \delta_1), Z), 4) \vdash$
$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, \delta_1(\delta_0, \delta_0), W), Z), 45) \vdash$
$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, \delta_1(\delta_0, Z), W), Z), 454) \vdash$
$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, \delta_1(Z, Z), W), Z), 4544) \vdash$
$\vdash (\delta_1(\delta_0, \delta_0(\delta_1, C, W), Z), 45443) \vdash$
$\vdash (\delta_1(\delta_0, \delta_0(W, C, W), Z), 454435) \vdash$
$\vdash (\delta_1(\delta_0, B, Z), 4544352) \vdash$
$\vdash (\delta_1(Z, B, Z), 45443524) \vdash$
$\vdash (A, 454435241)$

Tak więc automat $\mathfrak{A}_{DFT}$ zaakceptował wzorzec ($A \in F$) i rozpoznał go prawidłowo jako wzorzec I (ciąg produkcji 142534454 czytany od prawej do lewej).

Podobnie, jak w przypadku metody drzewowej opisanej w poprzednim punkcie zakładaliśmy dwa, te same, ograniczenia (na postać zbioru produkcji oraz sposób analizy – poruszania się po drzewie).

Zauważmy jeszcze, że algorytm analizy faktur zaprezentowaną metodą będzie taki sam, jak poprzednio. Jedynie w treści procedur **transfunc** i **replace** należy dokonać drobnych modyfikacji związanych z brakiem etykiet krawędziowych w opisie nawiasowym drzewa T.

12. METODY GRAFOWE

Jak wspomniano w rozdziale 9, gramatyki grafowe są mocniejszym narzędziem opisu obrazów, niż gramatyki ciągowe lub drzewowe. Dlatego też, użycie grafów do *opisu* dwu- lub trójwymiarowych obrazów jest powszechnie spotykane w literaturze [34]. Natomiast wykorzystanie ich do *rozpoznawania* obrazów nie jest tak powszechne. Spowodowane jest to trudnościami związanymi z analizą syntaktyczną gramatyk grafowych, dokładniej mówiąc, ze złożonością obliczeniową problemu analizy syntaktycznej, który dla zdecydowanej większości klas gramatyk jest NP-zupełny [34].

W rozdziale przedstawimy dwie, znane z literatury, syntaktyczne metody grafowe rozpoznawania obrazów: metodę parsingu ekspansywnych języków grafowych [35] oraz metodę parsingu dla gramatyki grafowej klasy $ETL(1)$ [36,37].

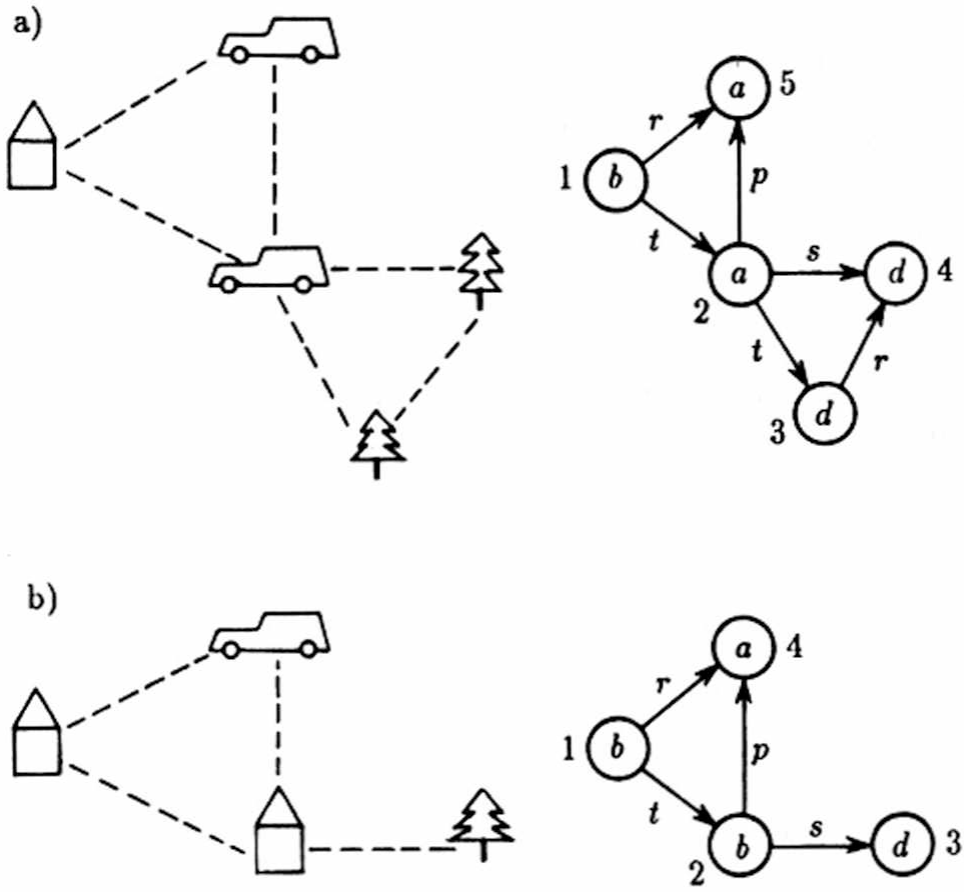
12.1. Parsing ekspansywnych języków grafowych

Przyjmijmy, że zbiory składowych pierwotnych reprezentujących obiekty scen i relacje pomiędzy obiektami są zilustrowane rysunkiem 11.1a. Rozważmy dwie sceny znajdujące się na rysunku 12.1. Sceny te możemy reprezentować w jednoznaczny sposób przez grafy klasy Ω([1]). Zapiszmy oba grafy Ω za pomocą ich opisów charakterystycznych.

I scena:	b_1	a_2	d_3	d_4	a_5
	2	3	1	0	0
	tr	tsp	r	–	–
	25	345	4	–	–

([1]) Pojęcia z teorii gramatyk grafowych używane w tym rozdziale zostały formalnie zdefiniowane w Dodatku 4.

II scena:	b_1	b_2	d_3	a_4
	2	2	0	0
	tr	sp	–	–
	24	34	–	–



Rys. 12.1. Dwie przykładowe sceny i ich reprezentacje grafowe

Zdefiniujmy następnie ekspansywną gramatykę grafową $\mathfrak{G}_{EXP}$ generującą obie sceny. Na jej bazie skonstruujemy algorytm parsera rozpoznającego sceny:

$$\mathfrak{G}_{EXP} = (N, \Sigma, \Gamma, \mathfrak{P}, S),$$

gdzie:

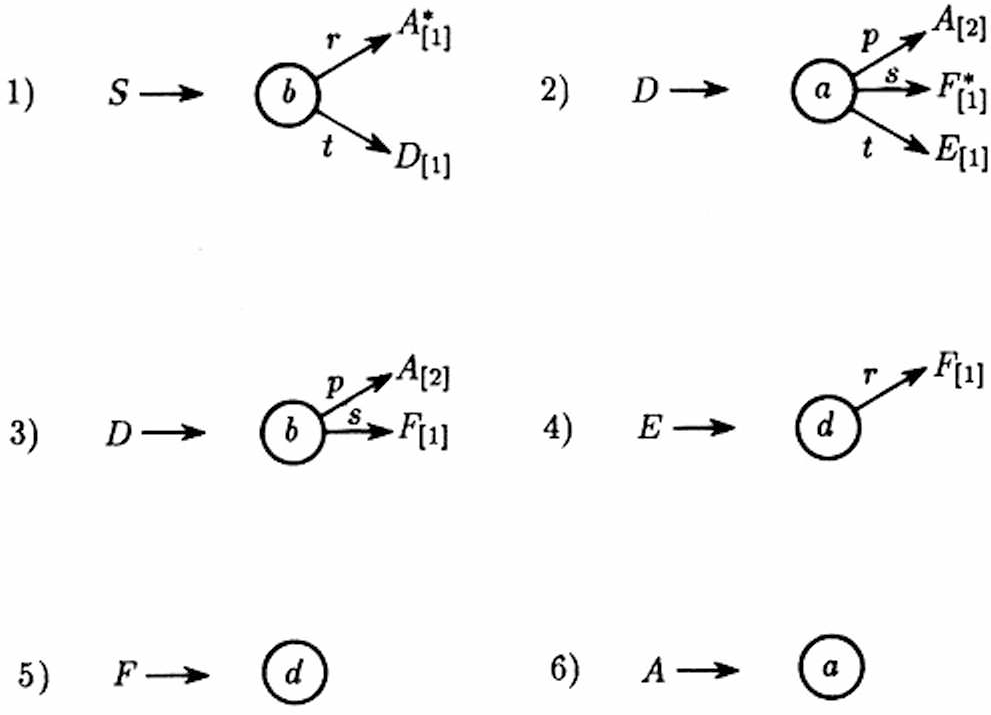
$N = \{S, A, D, E, F\}$,

$\Sigma = \{a, b, d\}$,

$\Gamma = \{r, t, p, u, s\}$,

$\mathfrak{P}$: (1) $S \to b\,tD\,rA^*$, (2) $D \to a\,tE\,sF^*pA$,
(3) $D \to b\,sF\,pA$, (4) $E \to d\,rF$,
(5) $F \to d$, (6) $A \to a$.

Zbiór produkcji $\mathfrak{P}$ został zilustrowany na rysunku 12.2 (cyfry w nawiasach kwadratowych oznaczają numery porządkowe przypisane nieterminalom prawych stron produkcji), natomiast generacja sceny I (według reguł opisanych w Dodatku 4), znajduje się na rysunku 12.3. Cyfry pod strzałkami oznaczają numery stosowanych produkcji, a napis ET oznacza zastosowanie transformacji osadzenia. Zauważmy, że „odziedziczenie" przez wierzchołek A krawędzi o etykiecie r po wierzchołku o etykiecie A^* jest możliwe, ponieważ numer porządkowy wierzchołka A (tzn. 2) jest wielokrotnością numeru porządkowego wierzchołka A^* (tzn. 1). Scenę II generujemy poprzez zastosowanie produkcji o numerach: 1, 3, 5, 6. Zatem wypisanie na wyjściu sekwencji produkcji 124546 po analizie przez parser jakiejś sceny oznacza, że jest to scena I, a 1356 – scena II.

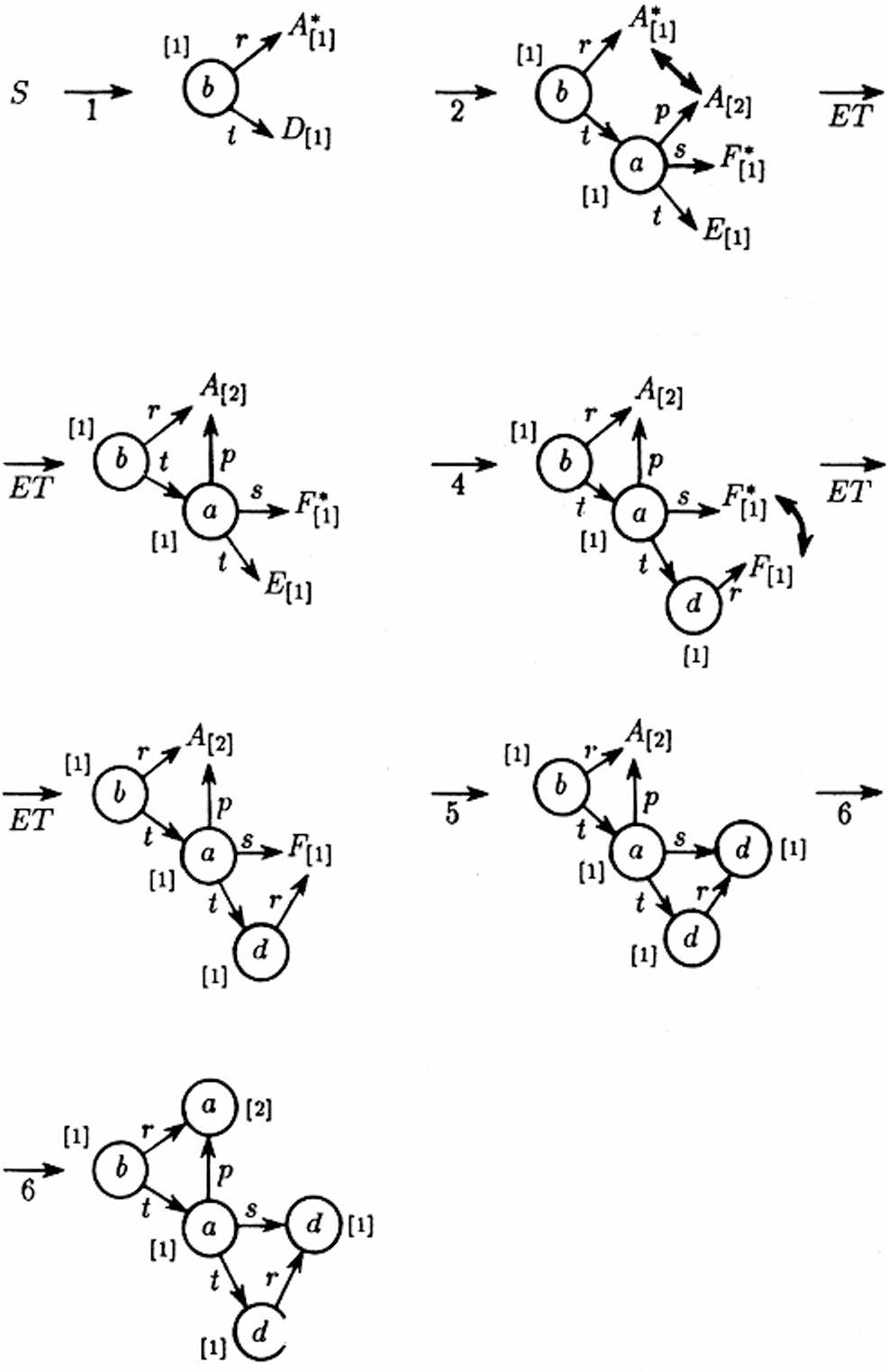


Rys. 12.2. Zbiór produkcji ekspansywnej gramatyki grafowej

Nim przedstawimy formalny zapis algorytmu dwuprzebiegowego parsera dla ekspansywnej gramatyki grafowej, zaprezentujemy ideę jego działania, analizując scenę I. W pierwszym przebiegu, parser konstruuje dla każdego wierzchołka n_i analizowanego grafu g trójkę $I_i = [N_i, \alpha_i, l_i]$, gdzie: N_i jest nieterminalem lewej strony produkcji takiej, że n_i jest wierzchołkiem terminalnym jej prawej strony, l_i numerem tej produkcji, α_i jest ciągiem produkcji startujących w generacji grafu g z tych nieterminali lewej strony l_i-tej produkcji, które nie są oznaczone przez $*$. Konstrukcja trójek przebiega od wierzchołka o najwyższym indeksie do wierzchołka o indeksie 1. Trójka I_1 powinna być postaci $[S, \alpha_1, l_1]$, tzn. ostatnim uzyskanym nieterminalem powinien być symbol startowy gramatyki, a w przeciwnym razie sygnalizowany jest błąd.

Analizując wierzchołek o indeksie 5 (grafu z rys. 12.1) zaetykietowany przez a definiujemy $I_5 = [A, \lambda, 6]$, λ – symbol pusty, gdyż a o stopniu wyjściowym 0 redukuje się do A w produkcji szóstej oraz z grafu prawej strony tej produkcji nie startują żadne produkcje. Analogicznie $I_4 = [F, \lambda, 5]$ (redukcja d o stopniu wyjściowym 0 do F w produkcji piątej). Dla wierzchołka o indeksie 3 i etykiecie d, z którego wychodzi krawędź r – $I_3 = [E, 5, 4]$, ponieważ wierzchołek o etykiecie d, z którego wychodzi krawędź r redukuje się do E w produkcji czwartej, a z nieterminala prawej strony tej produkcji (F) może wystartować tylko produkcja piąta. Analiza wierzchołka drugiego – a_2 daje nam $I_2 = [D, 46, 2]$ (redukcja wierzchołka o etykiecie a, z którego wychodzą krawędzie t, s, p, do D w produkcji drugiej). Zauważmy, że ciąg produkcji startujących jest dwuelementowy, a nie trójelementowy (produkcja czwarta dla E i produkcja szósta dla A), gdyż nieterminalny wierzchołek F^* został wprowadzony jako element transformacji osadzenia (i w trakcie wywodu powinien połączyć się z jakimś wierzchołkiem o etykiecie F). Analizując pierwszy wierzchołek grafu g – b_1 definiujemy trójkę $I_1 = [S, 2, 1]$. W tym przypadku, badając produkcje startujące z nieterminala grafu prawej strony produkcji pierwszej – D (A^* pomijamy, jak poprzednio) mamy do wyboru dwie produkcje: drugą i trzecią. Na podstawie opisu charakterystycznego grafu g stwierdzamy, że krawędź etykietowana przez t (wchodząca do nieterminala D w grafie prawej strony produkcji) wchodzi do wierzchołka o indeksie 2 zaetykietowanego przez a. Para D – a determinuje zatem jednoznacznie produkcję drugą (a nie trzecią, którą determinuje para D – b). Tak więc otrzymaliśmy następujące trójki:

$$I_1 = [S, 2, 1], \qquad I_2 = [D, 46, 2],$$



Rys. 12.3. Przebieg generacji sceny I z rys. 12.1 za pomocą ekspansywnej gramatyki grafowej

$I_3 = [E, 5, 4]$, $I_4 = [F, \lambda, 5]$,
$I_5 = [A, \lambda, 6]$.

W drugim przebiegu parsera będziemy analizować wierzchołki grafu, startując od wierzchołka o indeksie 1 i kończąc na wierzchołku o największym indeksie oraz wspomagając się wygenerowanymi trójkami. Zauważmy, że analizując np. wierzchołek o indeksie 2 zapamiętamy, że startują z niego produkcje: czwarta i szósta. Jakkolwiek w następnym kroku faktycznie będziemy sprawdzać, czy można zastosować produkcję czwartą do wierzchołka o indeksie 3, to sprawdzenie, czy możemy zastosować produkcję szóstą nastąpi dużo później – w naszym przypadku na końcu – gdyż jest ona związana z wierzchołkiem o indeksie 5. Będziemy musieli zatem zapamiętać konieczność sprawdzenia aplikacji pewnych produkcji w późniejszych krokach analizy.

Niech zatem wektor $\pi(i)$, $i = 1, \ldots, m$, gdzie m jest liczbą wierzchołków grafu, służy do zapamiętania numerów produkcji stosowanych w kolejnych wierzchołkach, i jest indeksem aktualnie analizowanego wierzchołka, a trójki I_i są poprzednio wprowadzonej postaci $[N_i, \alpha_i, l_i]$. Oznaczmy przez k sumę liczby produkcji, które, w momencie analizy i-tego wierzchołka, już rozważyliśmy i liczby tych produkcji, które zapamiętaliśmy do rozważenia w przyszłości.

Zauważmy, że jeśli przez d_i oznaczymy długość ciągu produkcji startujących α_i, to kolejne k będziemy otrzymywać zwiększając je w każdym kroku o d_i, tzn. $k := k + d_i$. Zatem, jeśli w i-tym kroku analizy okaże się, że $k = i$, tzn. nie zapamiętaliśmy takich produkcji, które będą rozważane w przyszłości, to zapamiętujemy jedynie ciąg produkcji startujących z wierzchołka i-tego:

$$\pi(i+1) \ldots \pi(i+d_i) := \alpha_i.$$

Natomiast jeśli $k > i$, tzn. w poprzednich krokach zapamiętaliśmy produkcje do rozważenia w przyszłości (jest ich $k - i$), to musimy je przepakować na dalsze pozycje w wektorze π, czyli

$$\pi(i+1+d_i) := \pi(i+1),$$
$$\ldots$$
$$\ldots$$
$$\pi(k+d_i) := \pi(k),$$

i wtedy dopiero na zwolnionych wcześniejszych miejscach wektora π zapamiętujemy ciąg produkcji startujących z i-tego wierzchołka:

$$\pi(i+1)\cdots\pi(i+d_i) := \alpha_i.$$

Przed formalną prezentacją algorytmu, prześledźmy jeszcze trzy pierwsze kroki drugiego przebiegu parsera dla naszego przykładu.

Inicjalizacja: i := 1; $k := 1$.

1. Stwierdzenie, że I_1 jest postaci $[S, \alpha_1, l_1]$, S - symbol startowy, umożliwia prawidłowe rozpoczęcie analizy;
$\pi(1) := 1\,(l_1)$;
$\pi(2) := 2\,(l_2)$;
$k := 1 + 1 \quad (k + d_1) = 2$;
$i := i + 1 = 2$.

2. Stwierdzenie, że $l_i = \pi(i)$ ($l_2 = \pi(2)$, tzn. $2 = 2$) oznacza zgodność numeru produkcji jaką zastosowaliśmy do wygenerowania grafu w i-tym kroku (podczas pierwszego przebiegu) – l_i z numerem produkcji jaka powinna być zastosowana w i-tym kroku – $\pi(i)$; stwierdzamy, że $k = i$, więc
$\pi(3) := 4$;
$\pi(4) := 6$;
$k := 2 + 2 \quad (k + d_2) = 4$;
$i := 3$.

3. Stwierdzamy, że $l_i = \pi(i)$ (tzn. $4 = 4$, $k > i$ $(4 > 3)$, więc:
a) $\pi(5) := \pi(4)$ (tzn. $\pi(5) := 6$),
b) $\pi(4) := 5$;
$k := 4 + 1 \quad (k + d_3) = 5$;
$i := 4$.

Po przeanalizowaniu wszystkich wierzchołków otrzymujemy ciąg $\pi(1)$ $\pi(2) \ldots \pi(5) = 12456$, co oznacza rozpoznanie sceny I.

Algorytm parsera przedstawimy przyjmując wprowadzone oznaczenia oraz oznaczając przez maketriple(i, I_i) wywołanie procedury tworzącej trójkę I_i dla wierzchołka o indeksie i. Zmienne: *rec*, *list*, *tab*, oraz funkcja decide zostały zdefiniowane w rozdziale 10.

```
procedure ExpRec (var rec);
begin
    for i := m to 1 do maketriple(i,I_i);
    i := 1;
    k := 1;
    if I_1 ≠ [S, α_1, l_1] then rec := 'err';
    else
      begin
        π(1) := l_1;
        π(2)π(3)...π(d_1 + 1) := α_1;
        k := k + d_1;
        for i := 2 to m do
        if rec ≠ 'err' then
          begin
            if (I_i = [A, α_i, l_i] and l_i = π(i)) then
              begin
                if k = i then
                  begin
                    π(i + 1)...π(i + d_i) := α_i;
                    k := k + d_i
                  end
                else (k > i)
                  begin
                    for c := i + 1 to k do π(c + d_i) := π(c);
                    π(i + 1)...π(i + d_i) := α_i;
                    k := k + d_i
                  end
              end
              else
              if (I_i = [A, λ, l_i] and l_i ≠ π(i)) then rec := 'err';
      end;
    if rec ≠ 'err' then
      begin
        list := π(1)...π(m);
        rec := decide(list,tab)
      end;
end.
```

12.2. Parsing dla gramatyki grafowej klasy $ETL(1)$

Metodę tą zilustrujemy przykładem analizy trzech scen: dwóch (I oraz II) – rozważanych w poprzednim punkcie (rys. 12.1) oraz sceny III (rys. 12.4). Zanim zdefiniujemy reprezentacje tych scen w postaci grafów IE [36], musimy wprowadzić relację porządku w zbiorze etykiet krawędziowych Γ przedstawionym na rysunku 11.1. Zróbmy to, przykładowo, w następujący sposób:

$$p \leq r \leq s \leq t \leq u \leq v \leq x \leq y.$$

Przy tak ustalonym porządku otrzymujemy grafy IE przedstawione na rysunkach 12.5, 12.6 oraz 12.7 o następujących opisach charakterystycznych:

I scena (rys. 12.1a):

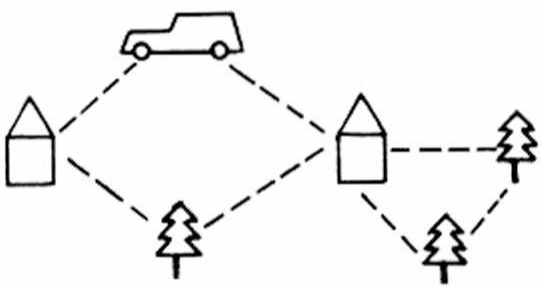
b_1	a_2	a_3	d_4	d_5
2	1	2	1	0
rt	u	st	v	–
23	3	45	5	–

II scena (rys. 12.1b):

b_1	a_2	b_3	d_4
2	1	1	0
rt	u	s	–
23	3	4	–

III scena (rys. 12.4):

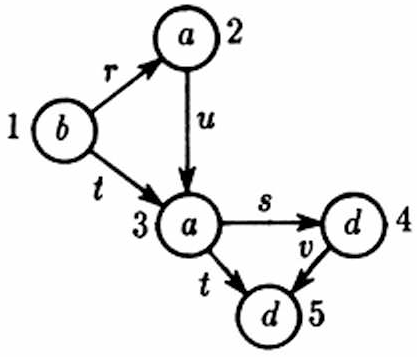
b_1	a_2	d_3	b_4	d_5	d_6
2	1	1	2	1	0
rt	t	r	st	v	–
23	4	4	56	6	–



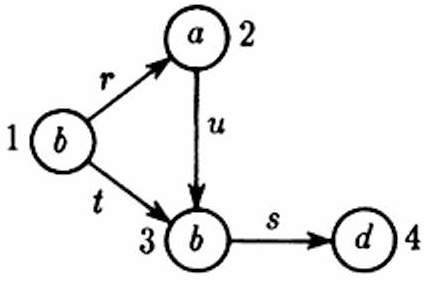
Rys. 12.4. Scena do analizy z wykorzystaniem gramatyki $ETL(1)$

Gramatyka grafowa klasy $ETL(1)$ generująca rozważane sceny jest postaci:

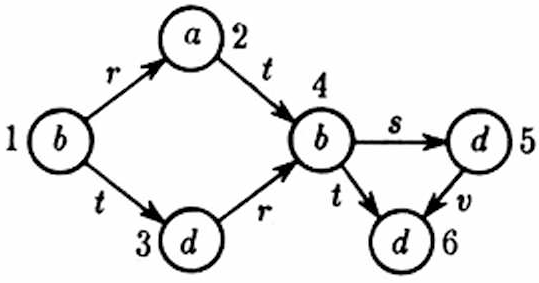
$$\mathfrak{G}_{ETL(1)} = (\Sigma, \Delta, \Gamma, \mathfrak{P}, Z),$$



Rys. 12.5. Graf dla sceny z rys. 12.1a



Rys. 12.6. Graf dla sceny z rys. 12.1b



Rys. 12.7. Graf dla sceny z rys. 12.4

gdzie:

$\Sigma = \{a, b, d, A, B\}$,

$\Delta = \{a, b, d\}$,

$\Gamma = \{r, s, t, u, v \mid r < s < t < u < v\}$.

Zbiór produkcji $\mathfrak{P}$, którego lewe i prawe strony są przedstawione na rysunku 12.8, zdefiniowano następująco:

(1) $A \rightarrow$

a_1	d_2	d_3	$C(t, in) = \{(a, b, t, in)\}$
2	1	0	$C(u, in) = \{(a, a, u, in)\}$
st	v	–	
23	3	–	

(2) $A \rightarrow$

b_1	d_2	$C(t, in) = \{(b, b, t, in)\}$
1	0	$C(u, in) = \{(b, a, u, in)\}$
s	–	
2	–	

(3) $A \rightarrow$

d_1	B_2	$C(t, in) = \{(d, b, t, in)\}$
1	0	$C(u, in) = \{(B, a, t, in)\}$
r	–	
2	–	

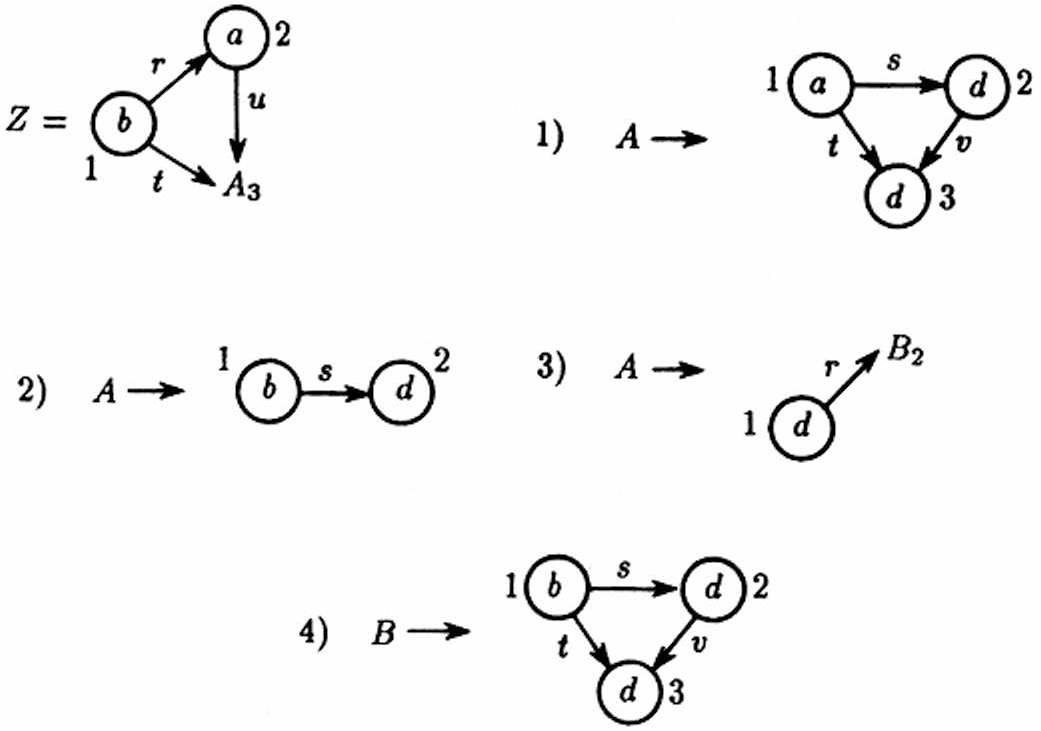
(4) $B \rightarrow$

b_1	d_2	d_3	$C(t, in) = \{(b, a, t, in)\}$
2	1	0	$C(r, in) = \{(b, d, r, in)\}$
st	v	–	
23	3	–	

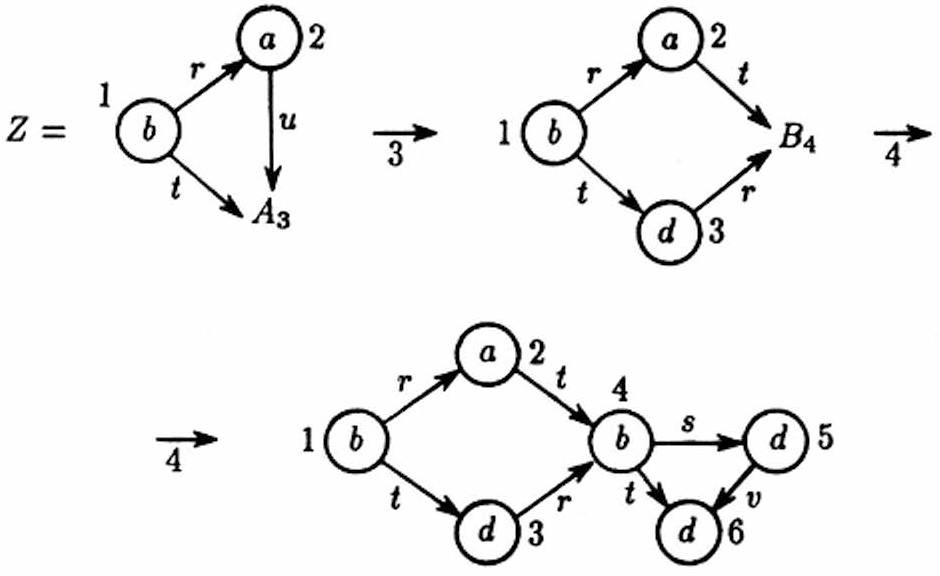
Graf startowy Z znajdujący się na rysunku 12.8 ma opis charakterystyczny:

b_1	a_2	A_3
2	1	0
rt	u	–
23	3	–

Jednoprzebiegowy parser typu generacyjnego (ang. *top-down*), którego algorytm przedstawimy na końcu tego punktu, dokonuje analizy syntaktycznej badając w każdym kroku opis charakterystyczny tylko jednego wierzchołka. Równocześnie konstruowany jest wywód, mający doprowadzić do wygenerowania analizowanego grafu (sceny). Podczas parsingu porównywane są wierzchołki analizowanego i wywodzonego grafów. Jeśli oba wierzchołki są terminalne, to badamy ich opisy charakterystyczne, a jeśli wierzchołek wywodzonego grafu jest nieterminalny, to szukamy takiej produkcji, po zastosowaniu której badane wierzchołki będą zgodne co do ich opisów. Numery produkcji użytych w trakcie parsingu są podstawą do zaklasyfikowania rozpoznawanej sceny. Rozważanym scenom odpowiadają następujące ciągi produkcji: I scena – 1, II scena – 2, III scena – 34.



Rys. 12.8. Produkcje gramatyki grafowej $ETL(1)$



Rys. 12.9. Analiza grafu (opis w tekście)

Rozważmy teraz następujący problem związany z analizą języków generowanych przez gramatyki $ETL(1)$. Załóżmy, że analizujemy graf g reprezentujący scenę III (rys. 12.4). Analizę rozpoczynamy od grafu startowego Z (rys. 12.9). Zauważmy, że jeśli analizujemy wierzchołki grafów g i Z indeksowane przez 2, to okaże się, że ich opisy charakterystyczne nie są zgodne. Wierzchołek grafu g indeksowany przez 2 jest opisany przez:

a_2
1
t
4

podczas gdy wierzchołek grafu Z indeksowany przez 2 jest opisany przez

a_2
1
u
3

Taka sytuacja wynika z faktu, że wierzchołek indeksowany przez 4 (do którego powinna wchodzić krawędź wychodząca z a_2) nie istnieje w grafie Z, natomiast krawędź u wychodząca z wierzchołka a_2 wchodzi do wierzchołka nieterminalnego indeksowanego przez 3. Tak więc, po zastosowaniu produkcji do wierzchołka A_3 sytuacja może się zmienić wskutek działania transformacji osadzenia. Dlatego zapamiętamy następującą trójkę (4,2,t), gdzie:

4 jest indeksem wierzchołka, do którego rozważana krawędź powinna wchodzić,

2 jest indeksem wierzchołka, z którego rozważana krawędź wychodzi,

t jest etykietą, którą powinna mieć rozważana krawędź (zgodnie z opisem charakterystycznym analizowanego grafu g).

Następnie analizujemy wierzchołek indeksowany przez 3. Szukamy produkcji postaci $A \rightarrow d\ldots$, ponieważ wierzchołek grafu g jest etykietowany przez d, a wierzchołek grafu Z jest etykietowany przez A. Taką produkcją jest produkcja trzecia i po jej zastosowaniu otrzymamy graf g_1 przedstawiony na rysunku 12.6b. Analiza wierzchołków indeksowanych przez 3 grafów g oraz g_1 wykazuje ich zgodność, w związku z czym możemy przejść

do analizy wierzchołków obu grafów indeksowanych przez 4. Po zastosowaniu produkcji czwartej do wierzchołka 4 grafu g_1 powinniśmy sprawdzić zgodność trójki (4,2,t) z sytuacją otrzymaną w wywodzonym grafie, który teraz jest postaci przedstawionej na rysunku 12.9. Po stwierdzeniu, że z wierzchołka indeksowanego przez 2 wychodzi krawędź etykietowana przez t wchodząca do wierzchołka o indeksie 4, możemy zapamiętaną trójkę usunąć z pamięci. Zauważmy jeszcze, że powinniśmy przeszukiwać zbiór zapamiętanych trójek przed każdym przejściem do analizy wierzchołka o indeksie $k+1$ sprawdzając, czy nie została wcześniej zapamiętana trójka postaci $(k,\ldots,\ldots)$.

Wprowadźmy teraz pojęcia, które pozwolą formalnie zdefiniować algorytm analizy syntaktycznej [37].

Niech będą dane grafy IE: g oraz g'. Dwa wierzchołki $n_k \in g$ i $n'_k \in g'$ mające ten sam indeks k nazywamy *wierzchołkami kontekstowo identycznymi*, jeśli mają takie same opisy charakterystyczne.

Dwa wierzchołki $n_k \in g$ i $n'_k \in g'$ mające ten sam indeks k oraz opisane przez

n_k,	n'_k (odpowiednio)
r	r'
$e_1 \ldots e_r$	$e'_1 \ldots e'_r$
$i_1 \ldots i_r$	$i'_1 \ldots i'_{r'}$

nazywamy *wierzchołkami potencjalnie kontekstowo identycznymi*, jeśli

1) $n_k = n'_k$,

2) $r = r'$,

3) jeśli dla każdego $j \in \{1,\ldots,r\}$ istnieje $k \in \{1,\ldots,r\}$ takie, że $i_j = i_k$, to $e_j = e_k$,

4) niech będą dane podciągi $\underline{i}_1,\ldots,\underline{i}_p$ oraz $\underline{i}'_1,\ldots,\underline{i}'_p$ ciągów $i_1,\ldots,i_r$ oraz $i'_1,\ldots i'_r$ (odpowiednio), które spełniają następujący warunek: dla żadnego $\underline{i}_j$, $j = 1,\ldots,p$ nie istnieje $i'_k, k = 1,\ldots,r$ takie, że $\underline{i}_j = i'_k$ oraz dla żadnego $\underline{i}'_j$, $j = 1,\ldots,p$ nie istnieje i_k, $k = 1,\ldots,r$ takie, że $\underline{i}_j = i_k$. Wtedy, dla każdego $j = 1,\ldots,p$: $\underline{i}_j$ jest etykietowany przez symbol terminalny, a $\underline{i}'_j$ przez symbol nieterminalny.

Wierzchołki i_j, $j = 1, \ldots, p$ nazywamy wierzchołkami potencjalnie kontekstowymi dla wierzchołka n_k, natomiast krawędzie wchodzące do wierzchołków i_j nazywamy krawędziami potencjalnie kontekstowymi dla wierzchołka n_k. Wierzchołki i'_j, $j = 1, \ldots, p$ nazywamy wierzchołkami quasi-kontekstowymi dla wierzchołka n_k.

Zauważmy, że wierzchołki indeksowane przez 2 grafów g oraz Z, w rozważanym przykładzie, są potencjalnie kontekstowo identyczne. Natomiast elementy zapamiętanej trójki postaci (q, k, e) mają następujące znaczenie:

q jest indeksem wierzchołka potencjalnie kontekstowego dla wierzchołka indeksowanego przez $k - n_k$,

e jest etykietą krawędzi potencjalnie kontekstowej dla wierzchołka n_k.

Trójkę o postaci (q, k, e) nazywamy opisem potencjalnie kontekstowej identyczności.

Przed prezentacją algorytmu parsera wprowadźmy następujące oznaczenia:

G – analizowany graf,

H – graf wywodzony (generowany) w trakcie parsingu,

Z – graf startowy gramatyki,

D – zbiór opisów potencjalnie kontekstowej identyczności,

$n(i)$ – etykieta wierzchołka grafu G indeksowanego przez i,

$n'(i)$ – etykieta wierzchołka grafu H indeksowanego przez i,

m – liczba wierzchołków analizowanego grafu.

Zdefiniujmy procedury i funkcje parsera:

choose(A, a, k) – jeśli istnieje produkcja, dla której A jest etykietą lewej strony oraz a jest etykietą wierzchołka pierwszego poziomu grafu prawej strony produkcji, to:

$k :=$ numer takiej produkcji;

w przeciwnym przypadku $k := 0$;

production(H, i, k) – zastosowanie k-tej produkcji dla i-tego wierzchołka grafu H ;

conid(G, H, i) – boolowska funkcja sprawdzająca, czy wierzchołki indeksowane przez i grafów G oraz H są kontekstowo identyczne;

pconid(G, H, i, D, rec) – procedura dołącza do zbioru D opis potencjalnie kontekstowej identyczności, jeśli wierzchołki indeksowane przez i grafów G oraz H są potencjalnie kontekstowo identyczne, w przeciwnym przypadku $rec :=' err'$;

check(D, H, i, rec) – jeśli istnieje opis potencjalnie kontekstowej identyczności $(q, k, e) \in D$ taki, że $q = i$, to procedura usuwa (q, k, e) ze zbioru D w przypadku zgodności opisu z sytuacją w grafie H, a w przypadku niezgodności $rec := 'err'$.

Zmienne: *rec*, *list*, *tab*, procedura **remember** oraz funkcja **decide** zostały zdefiniowane w rozdziale 11.

```
procedure ETLRec (var rec);
begin
      H := Z;
      for i := 1 to m do
      if rec ≠ 'err' then
        begin
          if n'(i) jest wierzchołkiem nieterminalnym then
            begin
              choose(n'(i), n(i), k);
              if k = 0 then rec := 'err'
              else      {żądana produkcja istnieje}
                begin
                  production(H, i, k);
                  remember(k)
                end;
            end;
          if not conid(G, H, i) then pconid(G, H, i, D, rec);
          check(D, H, i, rec);
        end;
      if rec ≠ 'err' then rec := decide(list,tab);
end;
```

Zakończymy ten rozdział kilkoma uwagami co do przedstawionych metod: w obu przypadkach algorytm analizy syntaktycznej ma złożoność

rzędu $O(n^2)$. Jakkolwiek obie metody zostały zdefiniowane dla potrzeb analizy scen, to ze względu na dużą moc opisową języków grafowych, zostały one wykorzystane również do innych celów: rozpoznawania pisma chińskiego (parsing ekspansywnych języków grafowych), analizy partytur orkiestrowych i rozpoznawania stanu zasobów sieci komputerowych w systemach alokacji (parsing dla gramatyki grafowej klasy $ETL(1)$).

Dodatek 1
Problem wyboru metryki w przestrzeni cech

Definiując w rozdziale 4 metody minimalnoodległościowe bazowano na istnieniu w przestrzeni cech X metryki ρ, której charakter nie był bliżej omawiany (por. wzór (22)). Obecnie zajmiemy się przydatnością konkretnych metryk. Najbardziej naturalna wydaje się zawsze metryka Euklidesowa, definiowana wzorem:

$$\rho_1(\underline{x}^\mu, \underline{x}^\eta) = \sqrt{\sum_{\nu=1}^{n}(x_\nu^\mu - x_\nu^\eta)^2}. \tag{D1.1}$$

Metryka ta odpowiada obiegowej definicji odległości i z tego powodu jest chętnie, odruchowo wręcz, stosowana w wielu praktycznie realizowanych algorytmach. Należy zdawać sobie sprawę, że metryka ta ma wiele wad. Wskazując niektóre z tych wad, możemy zaproponować inne warianty metryk.

Pierwsza wada wynika z ewentualnych różnic wymiarów poszczególnych składowych wektora x. Jeśli składowa x wyraża się – przykładowo – wartością rzędu 10^6, a pozostałe składowe – wartością rzędu 10^{-6}, to oczywiście wzór (D1.1) uzależnia odległość ρ_1 wyłącznie od wartości składowej x_ν przy całkowitym ignorowaniu pozostałych składowych. Najczęściej efekt taki jest całkowicie *niezgodny* z zamierzeniem twórcy metody, gdyż podważa celowość pomiaru pozostałych cech i niweluje wnoszoną przez nie informację. Jedyne rozsądne rozwiązanie polega wówczas na zastosowaniu mnożników normalizujących i uogólnionej metryki Euklidesowej o postaci:

$$\rho_2(\underline{x}^\mu, \underline{x}^\eta) = \sqrt{\sum_{\nu=1}^{n}[\lambda_\nu(x_\nu^\mu - x_\nu^\eta)]^2}. \tag{D1.2}$$

Mnożniki normalizujące λ mogą być w różny sposób związane z wartościami składowych wektora x_ν. Przykładowo można opierać je na przedziale zmienności

$$\lambda_\nu = \frac{1}{\max\limits_{\underline{x}\in U} x_{nu} - \min\limits_{\underline{x}\in U} x_{nu}}, \tag{D1.3}$$

lub na dyspersji

$$\lambda_\nu = \sqrt{\frac{N-1}{\sum\limits_{\underline{x}\in U}\left(x_\nu - \frac{1}{N}\sum\limits_{\underline{x}\in U} x_\nu\right)^2}}. \tag{D1.4}$$

Kolejny zarzut, jaki można postawić metryce ρ_1, jest związany z efektywnością obliczeniową. Operacje podnoszenia do kwadratu i pierwiastkowania, jakie w niej występują, wymagają długich czasów obliczeń. Prostsza pod tym względem jest metryka uliczna

$$\rho_3(\underline{x}^\mu, \underline{x}^\eta) = \sum_{\nu=1}^{n} | x_\nu^\mu - x_\nu^\eta |, \tag{D1.5}$$

bardzo sprawna obliczeniowo i dająca dobre rezultaty w przypadku zagadnień praktycznych, lub uogólniona metryka uliczna

$$\rho_4(\underline{x}^\mu, \underline{x}^\eta) = \sum_{\nu=1}^{n} \lambda_\nu | x_\nu^\mu - x_\nu^\eta |, \tag{D1.6}$$

Prosta jest także metryka Czebyszewa

$$\rho_5(x^\mu, x^\eta) = \max_{1\le\nu\le\eta} | x_\nu^\mu - x_\nu^\eta |, \tag{D1.7}$$

której obliczanie musi się jednak wiązać z normalizacją cech.

Metryki ρ_1, ρ_3 oraz ρ_5 stanowią szczególne przypadki metryki Minkowskiego

$$\rho_6(\underline{x}^\mu, \underline{x}^\rho) = \left[\sum_{\nu=1}^{n} | x_\nu^\mu - x_\nu^\eta |^t\right]^{\frac{1}{t}}, \tag{D1.8}$$

przy czym dla $t = 2$ mamy $\rho_6 = \rho_1$, dla $t = 1$ jest $\rho_6 = \rho_3$, zaś dla $t \to \infty$ również $\rho_6 \to \rho_5$. Dobierając wartości składnika t, można dość elastycznie dopasowywać metrykę ρ_6 do specyfiki konkretnego zadania rozpoznawania. Nie można jednak nie odnotować czasochłonności metryki ρ_6 w jej ogólnej postaci, wynikającej z maszynowego wykonywania operacji potęgowania przy wykładniku rzeczywistym.

Kolejnym defektem, wspólnym dla metryk $\rho_1 \div \rho_6$, jest fakt oparcia ich koncepcji na milczącym założeniu o *ortogonalności bazy przestrzeni cech* X. Założenie to w ogólnym przypadku jest niczym nie usprawiedliwione, gdyż wybór cech x_ν jest zwykle podyktowany możliwościami pomiarowymi i nie ma żadnych przesłanek, że tak uzyskane wartości x_ν wyznaczają ortogonalną bazę. Wprost przeciwnie, w większości praktycznych zadań można bez trudu wykazać, że poszczególne składowe wektora x są ze sobą skorelowane.

Rozważmy, jakie są tego praktyczne konsekwencje. Załóżmy, że składowa x_ν jest silnie skorelowana ze składowymi x_μ oraz x_η. Oznacza to, między innymi, że istnieje równanie regresji o postaci:

$$x_\nu = \beta_{\mu\nu} x_\mu + \beta_{\eta\nu} x_\eta + \beta_{o\nu} + \varepsilon_\nu, \tag{D1.9}$$

pozwalające wyznaczać wartości x_ν na podstawie wartości x_μ oraz x_η. Wartości współczynników regresji $\beta_{\mu\nu}$, $\beta_{\eta\nu}$ oraz $\beta_{o\nu}$ mogą być wyznaczone przy użyciu standardowych procedur analizy regresyjnej, zaś składnik ε_ν może być pomijalnie mały przy silnej korelacji pomiędzy x_ν a x_μ i x_η. Pominięcie składnika ε_ν we wzorze (D1.9) prowadzi jednak natychmiast do wniosku, że wszystkie wartości x_ν leżą na pewnej prostej (wyznaczonej przez $\beta_{\mu\nu}$, $\beta_{\mu\nu}$, $\beta_{o\nu}$) na płaszczyźnie $x_\mu x_\eta$. Jak z tego wynika, traktowanie x_ν jako zmiennej składowej odkładanej na osi prostopadłej do pozostałych jest zniekształceniem rzeczywistości.

Możliwe i celowe jest więc stosowanie innych metryk, na przykład metryki Mahalanobisa

$$\rho_7(x^\mu, x^\eta) = \sqrt{(\underline{x}^\mu - \underline{x}^\eta)^T \mathbf{T}^{-1} (\underline{x}^\mu - \underline{x}^\eta)}, \tag{D1.10}$$

w której obecność macierzy kowariancji cech $\mathbf{T}$ (patrz wzory (116) i (118)) prowadzi do „wyprostowania" nieortogonalnego układu współrzędnych na drodze apriorycznej *transformacji liniowej*. Warto zauważyć, że dla cech *zdekorelowanych* ρ_7 sprowadza się do metryki ρ_2, przy λ_ν określonym wzorem (D1.4). Metryka Mahalanobisa jest przykładem metryki kwadratowej o ogólnym wzorze

$$\rho_8(x^\mu, x^\eta) = \sqrt{(x^\mu - x^\eta)^T H (x^\mu - x^\eta)}, \tag{D1.11}$$

która przy odpowiednim doborze macierzy transformacji H może uwzględniać różne własności przestrzeni cech. Przykładowo: dla H diagonalnej wracamy do metryki ρ_1, dla $H = \mathbf{T}^{-1}$ mamy metrykę ρ_7, zaś przyjęcie H zawierającej elementy niezerowe wyłącznie poza główną przekątną prowadzi do metryki Bhattacharyya, często wskazywanej jako optymalna w praktycznych zadaniach rozpoznawania i grupowania cech [3].

Dodatek 2

Dowód twierdzenia o zbieżności procesu uczenia dla aproksymacyjnej metody rozpoznawania obrazów

W rozdziela 6 przytoczono metodę uczenia (wzory (66) i (67)) pozwalającą wyznaczać zestawy wag([1]) V, za pomocą których funkcje przynależności o postaci (58) pozwalają bezbłędnie rozpoznawać wszystkie obiekty. Przytoczono tam również twierdzenie głoszące, że metoda ta gwarantuje uzyskanie rozwiązania po skończonej liczbie kroków. Twiedzenie to (w różnych wariantach i odmianach) jest dyskutowane w niemal wszystkich monografiach, dotyczących problemów rozpoznawania, jednak jego znaczenie i waga skłaniają do przytoczenia go także w tej książce, szczególnie, że udało się opracować wyjątkowo krótką wersję tego dowodu.

Teza ograniczona do przypadku dwóch klas – założenia

Na początek rozważymy przypadek dychotomii ($L = 2$). Dla tego przypadku zamiast dwóch funkcji przynależności

$$C^1(\underline{\tilde{x}}) = \sum_{\nu=0}^{n} V_\nu^1 \tilde{x}_\nu, \tag{D2.1}$$

$$C^2(\underline{\tilde{x}}) = \sum_{\nu=0}^{n} V_\nu^2 \tilde{x}_\nu \tag{D2.2}$$

oraz reguły majoryzacji

$$F^k = i \Leftarrow C^i(\underline{\tilde{x}}^k) > C^{i+1}(\underline{\tilde{x}}^k) \tag{D2.3}$$

można rozważać funkcję rozdzielającą

$$C^{12}(\underline{\tilde{x}}) = C^1(\underline{\tilde{x}}) - C^2(\underline{\tilde{x}}) = \sum_{\nu=0}^{n} (V_\nu^1 - V_\nu^2)\tilde{x}_\nu \tag{D2.4}$$

([1]) Rozważany jest tu przypadek funkcji liniowych, ale jak wskazano w rozdziale 6, przypadek nielinowy może być rozważany jako złożenie algorytmu liniowego i nieliniowej transformacji układu współrzędnych.

oraz regułę dyskryminacji znaku

$$p^k = \begin{cases} 1, & \text{gdy } c^{12}(\tilde{\underline{x}}^k) \geq 0, \\ 2, & \text{w przeciwym przypadku.} \end{cases} \tag{D2.5}$$

Dokonując korekty ciągu uczącego według następujących reguł:

$$U = \{\langle \underline{x}^k, i^k \rangle\} \rightarrow U' = \{\hat{\underline{x}}^k\} \tag{D2.6}$$

$$\underline{p}^k = \begin{cases} \tilde{\underline{x}}, & \text{gdy } i^k = 1, \\ -\tilde{\underline{x}}, & \text{gdy } i^k = 2, \end{cases} \tag{D2.7}$$

otrzymujemy w miejsce (D2.4) i (D2.5) jeszcze prostszą zależność

$$C^{12}(\tilde{\underline{x}}^k) > 0 \tag{D2.8}$$

dla wszystkich *poprawnie* rozpoznawanych obiektów ciągu U'.

Korekta ciągu uczącego spowodowała więc zamianę problemu *liniowej rozdzielności* ((D2.3) lub (D2.5)) na problem *liniowej zwartości* (D2.8). Równocześnie uproszczeniu uległa reguła uczenia w stosunku do wzorów (66) i (67), gdyż można ją teraz zapisać jako

$$V_\nu^{12}(k+1) = V_\nu^{12}(k) + \hat{x}_\nu^k, \tag{D2.9}$$

dla każdego błędnie sklasyfikowanego obiektu $\hat{\underline{x}}^k$. Ponieważ obiekty ciągu uczącego, które nie powodują błędów, nie wpływają na tok procesu uczenia, przeto mogą być z dalszych rozważań *usunięte*. Będziemy więc zakładali, *że każdy pokazany obiekt* $\hat{\underline{x}}^k$ *jest błędnie klasyfikowany* (to znaczy $\sum_{\nu=0}^{n} V_\nu^{12}(k)\hat{x}_\nu^k < 0$) i wykonywana jest korekta (D2.9). Załóżmy także dla prostoty, że

$$V_0^{12}(1) = V_1^{12}(1) = \cdots = V_n^{12}(1) = 0. \tag{D2.10}$$

Wówczas oczywiście

$$V_\nu^{12}(k+1) = \sum_{\mu=1}^{k} \hat{x}_\nu^\mu, \tag{D2.11}$$

ponieważ każdy pokaz był związany z korektą (D2.9).

TWIERDZENIE. *Przy wyżej sformułowanych oznaczeniach oraz przy dodatkowym założeniu, że istnieje wektor wag* $\hat{V}_\nu$; $\nu = 0, \ldots, n$, *gwarantujący poprawne rozpoznawania wszystkich elementów ciągu uczącego, proces uczenia dany wzorem (D2.11) jest zbieżny do wektora* $\hat{V}_\nu$; $\nu = 0, \ldots, n$, *oraz osiąga go w skończonej liczbie kroków.*

DOWÓD. Wprowadźmy idealny (docelowy) zbiór wag $\hat{V}_\nu$; $\nu = 0, \ldots, n$, przy czym oczywiście przyjmujemy, że zbiór ten gwarantuje prawidłowe rozpoznawanie wszystkich elementów ciągu uczącego

$$\forall \underline{\hat{x}}^k \in U' \left[\sum_{\nu=0}^{n} \hat{V}_\nu \hat{x}_\nu^k > 0 \right]. \tag{D2.12}$$

Określmy wartość pomocniczą

$$\eta_1 = \min_{\underline{\hat{x}} \in U'} \sum_{\nu=0}^{n} \hat{V}_\nu \hat{x}_\nu^k. \tag{D2.13}$$

Na podstawie (D2.12) $\eta_1 > 0$.

Obliczymy teraz wartość iloczynu skalarnego $\underline{\hat{V}}\,\underline{V}^{12}(k-1)$, traktując zbiory wag jako wektory

$$\underline{\hat{V}}\,\underline{V}^{12}(k+1) = \sum_{\nu=0}^{n} \hat{V}_\nu V_\nu^{12}(k+1). \tag{D2.14}$$

Wykorzystując (D2.11), można zapisać:

$$\sum_{\nu=0}^{n} \hat{V}_\nu V_\nu^{12}(k+1) = \sum_{\mu=1}^{k} \left[\sum_{\nu=0}^{n} \hat{V}_\nu \hat{x}_\nu^\mu \right]. \tag{D2.15}$$

Ale z (D2.13) wynika, że

$$\sum_{\nu=0}^{n} \hat{V}_\nu V_\nu^{12}(k+1) \geq k\eta_1, \tag{D2.16}$$

zaś z nierówności Cauchy'ego-Schwarza można oszacować:

$$|\,\underline{V}^{12}(k+1)\,|^2 = \sum_{\nu=0}^{n} \left[V_\nu^{12}(k+1)\right]^2 \geq \frac{\sum_{\nu=0}^{n} \left[\hat{V}_\nu V_\nu^{12}(k+1)\right]^2}{\sum_{\nu=0}^{n} (\hat{V}_\nu)^2}, \tag{D2.17}$$

przeto wykorzystując (D2.16), otrzymujemy

$$| \underline{V}^{12}(k+1) |^2 \geq \frac{(\eta_1)^2}{\sum_{\nu=0}^{n} (\hat{V}_\nu)^2} k^2, \tag{D2.18}$$

co oznacza, że w trakcie uczenia długość wektora wag rośnie szybciej niż kwadrat liczby pokazów. Równocześnie jednak obowiązuje zależność (D2.9), którą zapiszemy w formie:

$$V_\nu^{12}(\mu+1) = V_\nu^{12}(\mu) + \hat{x}_\nu^\mu, \tag{D2.19}$$

skąd można otrzymać

$$| V^{12}(\mu+1) |^2 = | \underline{V}^{12}(\mu) |^2 + 2 \sum_{\nu=0}^{n} V_\nu^{12}(\mu) \hat{x}_\nu^\mu + \sum_{\nu=0}^{n} (\hat{x}_\nu^\mu)^2. \tag{D2.20}$$

Ale z uwagi na organizację ciągu uczącego U'

$$\forall_{\mu \in [1,k]} \left[\sum_{\nu=0}^{n} V_\nu^{12}(\mu) \hat{x}_\nu^\mu < 0 \right], \tag{D2.21}$$

przeto

$$| V^{12}(\mu+1) |^2 - | \underline{V}^{12}(\mu) |^2 \leq \sum_{\nu=0}^{n} (\hat{x}_\nu^\mu)^2. \tag{D2.22}$$

Sumując nierówności (D2.22) dla $\mu = 1, 2, \ldots, k$, otrzymujemy

$$| \underline{V}^{12}(k+1) |^2 \leq \sum_{\mu=1}^{k} \sum_{\nu=0}^{n} (\hat{x}_\nu^\mu)^2. \tag{D2.23}$$

Wprowadzając nową wartość pomocniczą

$$\eta_2 = \max_{\underline{\hat{x}} \in U'} \sum_{\nu=0}^{n} (\hat{x}_\nu^\mu)^2, \tag{D2.24}$$

otrzymujemy oszacowanie *sprzeczne* z (D2.18)

$$| \underline{V}^{12}(k+1) |^2 \leq k \eta_2. \tag{D2.25}$$

Sprzeczność wzorów (D2.18) i (D2.25) można usunąć jedynie w ten sposób, że założy się ograniczoną zmienność k

$$k \leq \frac{\eta_2 \sum_{\nu=0}^{n} (\hat{V}_\nu)^2}{\eta_1}, \tag{D2.26}$$

co kończy dowód, gdyż wykazuje, że liczba pokazów musi być skończona, a po ich zaprezentowaniu osiągany jest stan pełnej poprawności procesu rozpoznawania.

Uogólnienie na przypadek wielu klas

Przeprowadzony dowód dotyczył szczególnego przypadku $L = 2$. Uogólnienie tego dowodu na przypadek $L > 2$ jest jednak bardzo proste. Problem majoryzacyjnej reguły wyboru jednej z L klas na podstawie funkcji przynależności

$$C^i(\underline{\tilde{x}}) = \sum_{\nu=0}^{n} V_\nu^i \tilde{x} \tag{D2.27}$$

można zamienić na problem liniowej zwartości (por. (D2.8))

$$C(\underline{\hat{x}}) = \sum_{\mu=0}^{L(n+1)-1} \hat{V}_\nu \hat{x}_\mu > 0 \tag{D2.28}$$

przez prosty zabieg, polegający na zastąpieniu $(n-1)$-elementowych wektorów ciągu uczącego $\underline{\tilde{x}}^k$ przez $L(n+1)$-elementowe wektory $\underline{\hat{x}}^k$, zbudowane według następującego schematu:

$$\hat{x}^k_{(i_k-1)(n+1)+\nu} = \tilde{x}^k_\nu; \quad \nu = 0, 1, \ldots, n, \tag{D2.29}$$

$$\hat{x}^k_{(F_k-1)(n+1)+\nu} = -\tilde{x}^k_\nu; \quad \nu = 0, 1, \ldots, n, \tag{D2.30}$$

Pozostałe elementy $\hat{x}^k_\mu = 0$([2]).

Wzory (D2.29) i (D2.30) są sprzeczne przy $i^k = F^k$, ale przypadki takie, podobnie jak w przedstawionym zadaniu $L = 2$, mogą być eliminowane z rozważań, gdyż nie powodują konieczności korekty. We wzorze (D2.28),

([2]) Szczegółową dyskusję przyjętych oznaczeń przeprowadzono w rozdziale 6.

obok wektora cech zmodyfikowanego w omówiony sposób, musi wystąpić $L(n+1)$-elementowy wektor wag. Oznaczymy go $\underline{\tilde{V}}$ i zbudujemy według schematu:

$$\tilde{V}(i-1)(n+1)+\nu = V^i; \quad i = 1,\ldots,L, \quad \nu = 0,\ldots,n. \tag{D2.31}$$

Łatwo zauważyć, że omówione podstawienia upraszczają znacznie schemat uczenia. Zamiast zależności (66), (67) i (68) można stosować analogiczną do wzoru (D2.9) regułę:

$$\tilde{V}_\mu(k+1) = \tilde{V}_\mu(k) + \hat{x}^k. \tag{D2.32}$$

Spostrzeżenie tego faktu pozwala przyjąć (z niezbędnymi modyfikacjami oznaczeń) ciąg zależności od (D2.10) do (D2.26) jako dowód przypadku $L > 2$. Jak z tego wynika, zbyteczny jest oddzielny dowód dla tego przypadku i teza twierdzenia o skuteczności procesu uczenia przenosi się w sposób natychmiastowy i naturalny na przypadek ogólny.

Dodatek 3

Podstawowe pojęcia teorii języków formalnych i automatów

Podstawowe pojęcia teorii języków formalnych zostały zdefiniowane w połowie lat pięćdziesiątych przez Noama Chomsky'ego w pracach związanych z badaniem matematycznych modeli gramatyk opisujących języki naturalne [21] (jakkolwiek gramatyki Chomsky'ego można traktować jako podsystemy Thue'go, norweskiego matematyka żyjącego w latach 1863-1922). Co prawda później okazało się, że zastosowanie teorii Chomsky'ego w lingwistyce ma stosunkowo ograniczony zasięg, ale stworzone przez niego pojęcia są niezwykle przydatne w tak podstawowych dziedzinach informatyki, jak: języki programowania, projektowanie kompilatorów, czy właśnie rozpoznawanie obrazów.

Wprowadźmy następujące pojęcia.

Alfabetem Σ nazywamy pewien skończony zbiór symboli.

Ciągiem (słowem, zdaniem) nad alfabetem Σ nazywamy każdy ciąg skończonej długości składający się z symboli alfabetu Σ.

Przykładowo, dla alfabetu $\Sigma = \{a, b\}$, następujące napisy są słowami nad Σ : $a, b, aa, ab, ba, bb, aaa, aab, \ldots$ Zbiór wszystkich ciągów nad alfabetem Σ oznaczamy przez Σ^+, tzn. $\Sigma^+ = \{a, b, aa, ab, bb, \ldots\}$ dla $\Sigma = \{a, b\}$.

Słowo nie składające się z żadnego symbolu jest określone jako *słowo puste* i jest oznaczane przez λ. Wprowadźmy oznaczenie $\Sigma^* = \Sigma^+ \cup \{\lambda\}$.

Przedstawiając definicję ciągowej gramatyki formalnej, dokonuje się zwykle następującego podziału na cztery klasy gramatyk według klasyfikacji Chomsky'ego.

Ciągową gramatyką formalną (gramatyką ciągową, gramatyką) nazywamy czwórkę:

$$\mathcal{G} = (\Sigma_N, \Sigma_T, P, S),$$

gdzie: Σ_N – zbiór symboli nieterminalnych, Σ_T – zbiór symboli terminalnych, P – zbiór produkcji (reguł przepisujących), S – symbol startowy, $S \in \Sigma_N$.

Zakładamy przy tym, że $\Sigma = \Sigma_N \cup \Sigma_T$ oraz $\Sigma_N \cap \Sigma_T = \emptyset$.

Gramatyką struktur frazowych nazywamy gramatykę, której produkcje są postaci:

$$\eta \longrightarrow \gamma, \quad \text{gdzie} \quad \eta \in \Sigma^* \Sigma_N \Sigma^*, \quad \gamma \in \Sigma^*.$$

Ciąg η składa się zatem z przynajmniej jednego symbolu nieterminalnego, przed (i po) którym może znaleźć się pewna liczba symboli alfabetu Σ. Ciąg η jest nazywany lewą stroną produkcji, natomiast γ – prawą stroną produkcji.

Gramatyką kontekstową nazywamy gramatykę, której produkcje są postaci:

$$\eta_1 A \eta_2 \longrightarrow \eta_1 \gamma \eta_2, \quad \text{gdzie} \quad \eta_1, \eta_2 \in \Sigma^*, \quad \gamma \in \Sigma^+, \quad A \in \Sigma_N.$$

Tak więc, gramatyka kontekstowa pozwala na zastąpienie symbolu nieterminalnego A niepustym ciągiem symboli γ wtedy, gdy A pojawiło się w kontekście ciągów η_1 i η_2.

Gramatyką bezkontekstową nazywamy gramatykę, której produkcje są postaci:

$$A \longrightarrow \gamma, \quad \text{gdzie} \quad A \in \Sigma_N, \quad \gamma \in \Sigma^+.$$

W przypadku gramatyki bezkontekstowej, nieterminal A jest zastępowany przez niepusty ciąg γ, niezależnie od kontekstu, w jakim się znajduje A.

Gramatyką prawostronnie (lewostronnie) regularną nazywamy gramatykę, której produkcje są postaci: $A \longrightarrow \gamma G$ (dla gramatyki lewostronnie regularnej: $A \longrightarrow G\gamma$) lub $A \longrightarrow \gamma$, gdzie $A, G \in \Sigma_N$, $\gamma \in \Sigma_T^+$.

W literaturze dotyczącej teorii języków formalnych dają się zauważyć rozbieżności pomiędzy postaciami przytaczanych definicji gramatyk według klasyfikacji Chomsky'ego. Głównie dotyczą one sposobu określania ciągu γ. Podane definicje przedstawiono opierając się na pracach Chomsky'ego (np. [21]).

Jeśli $\gamma \in \Sigma^+$, to przez γ^n oznaczymy ciąg powstały przez przepisanie ciągu γ n razy, a $|\gamma|$ oznacza długość (liczbę symboli z alfabetu Σ) w ciągu γ. Przykładowo, dla $\gamma = abc$: $\gamma^3 = abcabcabc$, $|\gamma| = 3$, $|\gamma^3| = 9$.

$\eta \longrightarrow \gamma$ oznacza (bezpośredni) krok wyprowadzający w gramatyce $\mathfrak{G}$, tzn. jeśli $\eta = \sigma_1 \chi \sigma_2$ i $\gamma = \sigma_1 \tau \sigma_2$, to istnieje produkcja $\chi \longrightarrow \tau \in P$.

$\eta \xrightarrow{*} \gamma$ oznacza wyprowadzenie w gramatyce $\mathfrak{G}$, tzn. istnieje taki ciąg $\eta_0, \eta_1, \ldots, \eta_k$, że $\eta_0 = \eta$, $\eta_k = \gamma$ oraz $\eta_i \longrightarrow \eta_{i+1} \in P$ dla $i = 0, 1, \ldots, k-1$. Oczywiście $\xrightarrow{*}$ jest zwrotnym przechodnim domknięciem $\longrightarrow$.

Językiem generowanym przez gramatykę $\mathfrak{G}$ nazywamy zbiór

$$L(\mathfrak{G}) = \{\gamma \mid \gamma \in \Sigma_T^* \quad \text{taki, że} \quad S \longrightarrow \gamma\}.$$

Zdefiniujmy teraz dogodną do analizy syntaktycznej podklasę gramatyk bezkontekstowych.

Dla $A \in \Sigma_N$: $first(A) = \{a \in \Sigma \mid A \xrightarrow{*} \gamma,\ \gamma \in \Sigma^+ \text{ oraz } \gamma = a\eta\}$. Tak więc, zbiór *first* dla nieterminala A zawiera wszystkie symbole, jakie mogą pojawić się na jego pozycji. Przykładowo, jeśli w gramatyce występują produkcje: $A \longrightarrow aB$, $A \longrightarrow Cb$, $C \longrightarrow eEa$, to $a, C, e \in first(A)$.

Niech $\eta \longrightarrow \gamma$ będzie bezpośrednim krokiem wyprowadzającym w gramatyce bezkontekstowej takim, że

$$\eta = \eta_1 A \eta_2, \quad \gamma = \eta_1 \tau \eta_2 \quad \text{i} \quad A \longrightarrow \tau \in P.$$

Jeśli $\eta_1 \in \Sigma_T^*$, to krok wyprowadzający nazywamy *lewostronną kanoniczną generacją* i oznaczamy $\eta \xrightarrow[L]{} \gamma$. Zwrotne i przechodnie domknięcie $\xrightarrow[L]{}$ nazywamy lewostronnym wyprowadzeniem i oznaczamy $\xrightarrow[L]{*}$. Innymi słowy, lewostronnym wyprowadzeniem nazywamy takie wyprowadzenie, gdzie w każdym kroku produkcję stosujemy dla najbardziej lewego nieterminala.

Bezkontekstową gramatykę $\mathfrak{G}$ nazywamy *gramatyką klasy* $LL(1)$ [22], [23], jeśli spełnione są dwa następujące warunki:

1) gramatyka nie zawiera produkcji o postaci $A \longrightarrow A\gamma$, $A \in \Sigma_N$, $\gamma \in \Sigma^*$,

2) jeśli istnieją dwa wyprowadzenia:

$$S \xrightarrow[L]{*} \gamma_1 A \gamma_2 \xrightarrow[L]{} \gamma_1 \sigma \gamma_2 \xrightarrow[L]{*} \gamma_1 \eta_1,$$

$$S \xrightarrow[L]{*} \gamma_1 A \gamma_2 \xrightarrow[L]{} \gamma_1 \tau \gamma_2 \xrightarrow[L]{*} \gamma_1 \eta_2$$

oraz zachodzi $first(\eta_1) = first(\eta_2)$, to $\sigma = \tau$.

Innymi słowy, z każdego nieterminala A do elementu zbioru $first(A)$ możemy pójść tylko jedną drogą (tzn. dla nieterminala A i elementów zbioru $first(A)$ mamy zdeterminowaną, w sensie stosowania produkcji, drogę).

Po wprowadzeniu klas gramatyk, których używamy do generacji ciągu uczącego w rozdziale 10, zdefiniujmy podstawowe klasy automatów, wykorzystywanych jako mechanizmy rozpoznawania obrazów.

Deterministycznym automatem o skończonej liczbie stanów nazywamy piątkę

$$\mathfrak{A} = (\Sigma_T, Q, \delta, q_0, F),$$

gdzie Σ_T – skończony zbiór symboli wejściowych, Q – skończony zbiór stanów, $\delta : Q \times \Sigma_T \longrightarrow Q$ – funkcja przejścia, q_0 – stan początkowy, $F \subseteq Q$ – zbiór stanów końcowych. Zakładamy przy tym, że

$$\delta(q, a\gamma) = \delta(\delta(q, a), \gamma), \quad \gamma \in \Sigma_T^*, \quad a \in \Sigma_T.$$

Przez $rp(\gamma)$ oznaczamy stan jaki osiąga automat $\mathfrak{A}$ po przeczytaniu ciągu γ, tzn.

$$rp(\gamma) = \delta(q_0, \gamma).$$

Zbiór słów akceptowanych przez automat $\mathfrak{A}$ definiujemy jako

$$T(A) = \{\gamma \in \Sigma_T^* \mid \delta(q_0, \gamma) \in F\}.$$

Automatem $\mathfrak{A}_{LL}$ klasy $LL(1)$ [22,23] nazywamy czwórkę

$$\mathfrak{A}_{LL} = (\Sigma_T, \Sigma, \delta, Z_0),$$

gdzie Σ_T – skończony zbiór symboli wejściowych, Σ – skończony zbiór symboli stosu, $\delta : \Sigma \times \Sigma_T \longrightarrow \{(\sigma, l), rem, acc, err\}$, $\sigma \in \Sigma^*$, $l \in \mathcal{N}$ – funkcja przejścia, Z_0 – symbol znajdujący się na dnie stosu $Z_0 \in \Sigma$.

W celu opisania działania automatu $\mathfrak{A}_{LL}$ klasy $LL(1)$ wprowadźmy pojęcie konfiguracji automatu. *Konfiguracją automatu* klasy $LL(1)$ nazywamy trójkę

$$(INPUT, STACK, OUTPUT),$$

gdzie $INPUT \in \Sigma_T^*$ – ciąg symboli, jakie znajdują się na wejściu automatu, $STACK \in \Sigma^+$ – ciąg symboli, jakie znajdują się na stosie, $OUTPUT$ –

ciąg numerów produkcji gramatyki, jakie zostały użyte podczas dotychczasowej analizy słowa.

Działanie automatu $\mathfrak{A}_{LL}$ klasy $LL(1)$ zdefiniujemy w następujący sposób:

1) konfiguracją początkową jest (η, SZ_0, λ), gdzie η jest słowem, które będziemy analizować, \$ - znacznikiem końca ($\$ \in \Sigma_T$), S - symbolem startowym gramatyki;

2) $(a\gamma, A_1\tau, \pi) \vdash (\gamma, \tau, \pi) \Leftrightarrow \delta(A_1, a) = rem$, $A_1 \in \Sigma$, $a \in \Sigma_T$, $\gamma \in \Sigma_T^*$, $\tau \in \Sigma^*$, π jest ciągiem numerów produkcji, $\vdash$ - oznacza przejście automatu z jednej konfiguracji do drugiej;

3) $(a\gamma, A_1\tau, \pi) \vdash (a\gamma, \sigma\tau, \pi i) \Leftrightarrow \delta(A_1, a) = (\sigma, i)$;

4) $(a\gamma, A_1\tau, \pi) \vdash acc \Leftrightarrow \delta(A_1, a) = acc$, gdzie acc jest konfiguracją rozpoznania ciągu η (i zakończenia działania automatu);

5) $(a\gamma, A_1\tau, \pi) \vdash err \Leftrightarrow \delta(A_1, a) = err$, gdzie err jest konfiguracją błędu - nierozpoznania ciągu η (i zakończenia działania automatu).

Z tego opisu wynika, że zbiór słów akceptowanych przez automat $\mathfrak{A}_{LL}$ klasy $LL(1)$ definiujemy jako

$$T(\mathfrak{A}_{LL}) = \{\gamma \in \Sigma_T^* \mid (\gamma\$, SZ_0, \lambda) \vdash * -(\$, Z_0, \pi)\},$$

gdzie π jest ciągiem numerów produkcji gramatyki $LL(1)$ prowadzących do wygenerowania γ.

Sekwencyjnym transdjuserem z wielokrotnym wejściem([1]) (ang. *sequential multiple entry transducer*) [24] nazywamy piątkę

$$ST = (Q, \Sigma_T, \Delta, \delta, Q_0),$$

gdzie Q - skończony zbiór stanów, Σ_T - skończony zbiór symboli wejściowych, Δ - skończony zbiór symboli wyjściowych, $\delta : Q \times \Sigma_T^* \longrightarrow Q \times \Delta^*$ - funkcja przejścia (przypadek deterministyczny), $Q_0 \subset Q$ - zbiór stanów początkowych.

([1]) Nie należy mylić sekwencyjnego transdjusera z wielokrotnym wejściem z bardziej znanym z teorii automatów deterministycznym transdjuserem o skończonej liczbie stanów (ang. *deterministic finite transducer*) [25].

Zapis $\delta(q, \gamma) = (q', \eta)$ oznacza, że po przeczytaniu ciągu $\gamma \in \Sigma_T^*$ z wejścia, transdjuser przechodzi ze stanu q do stanu q' i wypisuje ciąg $\eta \in \Delta^*$ na wyjście.

Dla $\gamma \in \Sigma_T^*$, ciąg $\eta = ST(\gamma)$ jest napisem wyjściowym wtedy i tylko wtedy, gdy istnieją ciągi

$$\gamma_1, \ldots, \gamma_k \in \Sigma_T^*, \quad \eta_1, \ldots, \eta_k \in \Delta^*, \quad q_1, \ldots, q_k \in Q$$

takie, że

1) $\gamma = \gamma_1 \ldots \gamma_k$,

2) $\eta = \eta_1 \ldots \eta_k$,

3) $\delta(q_i, \gamma_{i+1}) = (q_{i+1}, \eta_{i+1})$, $i = 0, 1, \ldots, k-1$, $q_0 \in Q_0$.

Dodatek 4

Wybrane pojęcia teorii języków drzewowych i grafowych

W dodatku znajdują się definicje pojęć wykorzystywanych w rozdziałach 11 i 12. Definicje drzewowych gramatyk ekspansywnych $\mathfrak{G}_e$ oparte są na pracy Brainerda [29], natomiast automaty $\mathfrak{A}_{DF}$ wprowadzimy tak, jak w [30], rozszerzając automat $\mathfrak{A}_{DF}$ do automatu z wyjściem.

Ekspansywną gramatyką drzewową generującą drzewa $\mathfrak{G}_T$ o nieskierowanych i niezaetykietowanych krawędziach, tzn. drzewa T nazywamy czwórkę

$$\mathfrak{G}_T = (\Sigma, r, P, Z),$$

gdzie $\Sigma = \Sigma_T \cup \Sigma_N$ – zbiór etykiet wierzchołkowych (terminalnych i nieterminalnych), P – zbiór produkcji o postaci:

$$A \rightarrow a(A_1 A_2 \ldots A_{r(a)}),$$

gdzie $A, A_1, \ldots, A_{r(a)} \in \Sigma_N$, $a \in \Sigma_T$, $a(A_1 A_2 \ldots A_{r(a)})$ jest drzewem T (zapisanym w postaci nawiasowej) o korzeniu a i $r(a)$ następnikach-liściach $A_1, A_2, \ldots, A_{r(a)}$, r jest funkcją przypisującą wierzchołkowi etykietowanemu przez a liczbę jego następników, Z jest skończonym zbiorem drzew startowych.

Ekspansywną gramatyką drzewową (lub gramatyką $\mathfrak{G}_{EDT}$) generującą drzewa o skierowanych i zaetykietowanych krawędziach, tzn. drzewa EDT (ang. Edge-labelled Directed Tree), nazywamy piątkę

$$\mathfrak{G}_{EDT} = (\Sigma, \Gamma, r, P, Z),$$

gdzie Σ, r, Z są takie jak w poprzedniej definicji, Γ – zbiór etykiet krawędziowych, P – zbiór produkcji o postaci:

$$A \rightarrow a(\tau_1 A_1 \tau_2 A_2 \ldots \tau_{r(a)} A_{r(a)}),$$

gdzie $A, A_1, A_2, \ldots, A_{r(a)} \in \Sigma_N$, $a \in \Sigma_T$, $\tau_1, \tau_2, \ldots, \tau_{r(a)} \in \Gamma$, natomiast $a(\tau_1 A_1 \tau_2 A_2 \ldots \tau_{r(a)} A_{r(a)})$ jest drzewem EDT (zapisanym w postaci nawiasowej) o korzeniu a i $r(a)$ następnikach-liściach $A_1, A_2, \ldots, A_{r(a)}$ połączonych z korzeniem $r(a)$ krawędziami zaetykietowanymi $\tau_1, \tau_2, \ldots, \tau_{r(a)}$ (odpowiednio) i skierowanymi od korzenia do liści.

Automatem $\mathfrak{A}_{DFT}$ z wyjściem nad zbiorem etykiet wierzchołkowych $\Sigma_T = \{a_1, \ldots, a_n\}$ rozpoznającym drzewa T nazywamy $(n+2)$-krotkę:

$$\mathfrak{A}_{EDT} = (Q, \delta_1, \ldots, \delta_n, F),$$

gdzie Q – skończony zbiór stanów, δ_k, $k = 1, \ldots, n$ – funkcja przejścia zdefiniowana następująco:

$$\delta_k : Q^{r(a_k)} \to Q \times \mathbf{N},$$

gdzie $a_k \in \Sigma_T$, $\mathbf{N}$ – zbiór liczb naturalnych, $F \subseteq Q$ – zbiór stanów końcowych.

Zdefiniujmy teraz funkcję rp (patrz Dodatek 3) dla automatu $\mathfrak{A}_{DFT}$:

1) Dla wierzchołka $a \in \Sigma_T$ będącego liściem (tzn. nie posiadającego następników – $r(a) = 0$):

$$rp(a) = (A, i) \Leftrightarrow \delta_a = (A, i),$$

gdzie i jest numerem produkcji, na bazie której została zdefiniowana δ_a, wypisywanym na wyjście.

2) Dla wierzchołka $a \in \Sigma_T$ posiadającego $r(a)$ następników, które są korzeniami poddrzew $y_1, \ldots, y_{r(a)}$

$$rp(a(y_1 \ldots y_{r(a)})) = (A, i) \Leftrightarrow$$
$$\exists A_1, \ldots, A_{r(a)} \in Q : \delta_a(A_1, \ldots, A_{r(a)}) = (A, i)$$

oraz $rp(y_k) = (A_k, j(k))$, $k = 1, \ldots r(a)$, gdzie i jest numerem produkcji, na bazie której została zdefiniowana $\delta_a(A_1, .., A_{r(a)})$, natomiast $j(k)$ jest numerem pierwszej produkcji generującej drzewo y_j (numery produkcji są wypisywane na wyjście), czyli

$$rp(a(y_1 \ldots y_{r(a)})) = \delta_a(rp(y_1), \ldots, rp(y_{r(a)})).$$

Automatem $\mathfrak{A}_{DFEDT}$ z wyjściem nad zbiorem etykiet wierzchołkowych $\Sigma_T = \{a_1, \ldots, a_n\}$ *i zbiorem etykiet krawędziowych* Γ rozpoznającym drzewa EDT nazywamy $(n+2)$-krotkę:

$$\mathfrak{A}_{DFEDT} = (Q, \delta_1, \ldots, \delta_n, F),$$

gdzie Q i F – jak w poprzedniej definicji, δ_k, $k = 1, \ldots, n$, jest funkcją przejścia zdefiniowaną następująco:

$$\delta_k : (\Gamma Q)^{r(a_k)} \to Q \times \mathbf{N},$$

gdzie $a_k \in \Sigma_T$, $\mathbf{N}$ – zbiór liczb naturalnych, $\Gamma Q = \{\tau A \mid \tau \in \Gamma, A \in Q\}$.

Funkcję rp dla automatu $\mathfrak{A}_{DFEDT}$ zdefiniujemy analogicznie jak dla automatu $\mathfrak{A}_{DFT}$

1) dla wierzchołka $a \in \Sigma_T$ będącego liściem

$$rp(a) = (A, i) \Leftrightarrow \delta_a = (A, i),$$

gdzie i – numer produkcji, na bazie której została zdefiniowana δ_a, wypisywany na wyjście,

2) dla wierzchołka $a \in \Sigma_T$ posiadającego $r(a)$ następników połączonych z wierzchołkiem a krawędziami $\tau_1, \ldots, \tau_{r(a)}$ i będących korzeniami poddrzew $y_1, \ldots, y_{r(a)}$

$$rp(a(\tau_1 y_1 \ldots \tau_{r(a)} y_{r(a)})) = (A, i) \Leftrightarrow$$
$$\exists A_1, \ldots, A_{r(a)} \in Q : \delta_a(\tau_1 A_1, \ldots \tau_{r(a)} A_{r(a)}) = (A, i)$$

oraz $rp(y_k) = (A_k, j(k))$, $k = 1, \ldots, r(a)$ gdzie $i, j(k)$ są takie jak w analogicznej definicji, czyli

$$rp(a(\tau_1 y_1 \ldots \tau_{r(a)} y_{r(a)})) = \delta_a(\tau_1 rp(y_1), \ldots, \tau_{r(a)} rp(y_{r(a)})).$$

Konfiguracją automatu drzewowego z wyjściem (w obu przypadkach) nazywamy parę $(INPUT, OUTPUT)$, gdzie $INPUT$ – ciąg symboli opisujący analizę drzewa zapisanego w postaci nawiasowej, $OUTPUT$ – ciąg numerów produkcji, jakie zostały użyte podczas dotychczasowej analizy drzewa.

Konfiguracją początkową jest (η, λ), gdzie η jest zapisem drzewa w postaci nawiasowej, λ – słowem pustym, a konfiguracją końcową jest (A, τ), gdzie τ jest ciągiem numerów produkcji gramatyki drzewowej prowadzącym do wygenerowania drzewa η, $A \in F$ (zbiór stanów końcowych). Zatem zbiór drzew akceptowalnych przez automat drzewowy $\mathfrak{A}_{DFT}$ (w obu przypadkach) określimy jako:

$$T(\mathfrak{A}_{DFT}) = \{\eta \mid (\eta, \lambda) \vdash\!\!\!\!- * -\!\!\!\!\vdash (A, \tau)\}.$$

Wprowadzimy teraz definicję grafu klasy EDG używanego do reprezentacji obrazów strukturalnych [33].

Skierowanym, wierzchołkowo i krawędziowo etykietowanym grafem (grafem EDG – ang. Edge-labelled Directed Graph) nazywamy piątkę

$$H = (V, E, \Sigma, \Gamma, \phi),$$

gdzie V – skończony, niepusty zbiór wierzchołków, Σ – skończony, niepusty zbiór etykiet wierzchołkowych, Γ – skończony, niepusty zbiór etykiet krawędziowych, E – zbiór krawędzi o postaci (v, λ, w), gdzie $v, w \in V$, $\lambda \in \Gamma$, $\phi : V \to \Sigma$ – funkcja etykietowania wierzchołków.

Definicje dotyczące grafów klasy Ω i rodziny ekspansywnych gramatyk grafowych zostały sformułowane na podstawie pracy [35].

Niech $H = (V, E, \Sigma, \Gamma, \phi)$ będzie grafem EDG o m wierzchołkach, którym przypisano indeksy $1, \ldots, m$. *Etykietowaną macierzą przyległości* dla H nazywamy macierz $A = [a_{ij}]_{m \times m}$ taką, że

a) $a_{ij} = \beta$, jeśli $(v_i, \beta, v_j) \in E$, gdzie $v_i(v_j)$ jest wierzchołkiem o indeksie $i(j)$,

b) $a_{ij} = 0$, w przeciwnym przypadku.

Funkcją maksymalnego poprzednika nazywamy funkcję

$$maxind : \{1, \ldots, m\} \to \{1, \ldots, m\}$$

taką, że $maxind(k) = l$, gdy

1) z wierzchołka o indeksie l – v_l wychodzi krawędź wchodząca do wierzchołka o indeksie k – v_k,

2) wierzchołek v_l jest maksymalnym (w sensie indeksu) wierzchołkiem o własności przedstawionej w punkcie 1.

Spójny graf EDG H o macierzy przyległości $A = [a_{ij}]_{mxm}$ nazywamy *grafem klasy* Ω (grafem Ω) wtedy i tylko wtedy, gdy jego wierzchołki można zindeksować tak, że spełnione są następujące warunki:

1) $a_{ij} = 0 \; \forall i \geq j$,

2) dla każdego a_{ij} takiego, że $a_{ij} = 1$ oraz $i \neq maxind(j)$, istnieje uporządkowany ciąg indeksów $i < j_1 < j_2 < \cdots < j_r < j$ taki, że

$$i = maxind(j_1),\ j_1 = maxind(j_2), \ldots,\ j_r = maxind(j).$$

Niech n_k będzie wierzchołkiem o indeksie k i etykiecie n_k grafu H. Czwórkę

$$(n_k, \quad r, \quad e_1 \ldots e_r, \quad i_1 \ldots i_r),$$

gdzie r – stopień wyjściowy n_k (tzn. liczba krawędzi wychodzących z n_k), $i_1 \ldots i_r$ – uporządkowany rosnąco ciąg indeksów wierzchołków, do których wchodzą krawędzie wychodzące z n_k, $e_1 \ldots e_r$ – ciąg etykiet krawędziowych odpowiadających krawędziom wchodzącym do wierzchołków o indeksach $i_1, \ldots, i_r$, nazywamy *opisem charakterystycznym wierzchłka* n_k. Ciąg opisów charakterystycznych kolejnych wierzchołków grafu H nazywamy *opisem charakterystycznym grafu* H.

Ekspansywną gramatyką grafową $\mathcal{G}_{EXP}$ nazywamy piątkę

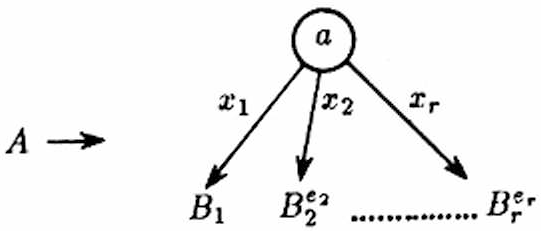
$$\mathcal{G}_{EXP} = (N, \Sigma, \Gamma, P, S),$$

gdzie N – skończony zbiór nieterminalnych etykiet wierzchołkowych, Σ – zbiór terminalnych etykiet wierzchołkowych, Γ – skończony zbiór etykiet krawędziowych, $S \in N$ – symbol startowy, P – skończony zbiór produkcji o jednej z dwóch postaci:

a) o postaci zredukowanej $A \rightarrow a$, gdzie $A \in N$, $a \in \Sigma$,

b) o postaci ekspansywnej przedstawionej na rysunku D4.1 i zapisywanej skrótowo:

$$A \rightarrow a x_1 B_1 \; x_2 B_2^{e^2} \ldots x_r B_r^{e^r},$$



Rys. D4.1. Ekspansywna postać produkcji

gdzie $a \in \Sigma$, $A, B_i \in N$, $x_i \in \Gamma$, $e_i \in \{*, \lambda\}$, $i = 1, \ldots, r$, są operatorami osadzenia, λ jest symbolem pustym.

Numerami porządkowymi nieterminali prawej strony produkcji o postaci ekspansywnej $A \to a\ x_1 B_1\ x_2 B_2^{e_2} \ldots x_r B_r^{e_r}$ nazywamy liczby naturalne przypisane do B_i, $i = 2, \ldots, r$, w następujący sposób:

1) B_1 ma numer porządkowy 1,

2) jeśli $e_i = \lambda$, $i = 2, \ldots, r$, to B_i ma numer porządkowy o 1 większy niż B_{i-1},

3) jeśli $e_i = *$, $i = 2, \ldots, r$, to B_i ma numer porządkowy taki sam jak B_{i-1}.

Przykładowe produkcje ekspansywnej gramatyki grafowej z przypisanymi numerami porządkowymi znajdują się na rys. 12.2.

Wywód w ekspansywnej gramatyce grafowej $\mathcal{G}_{EXP} = (N, \Sigma, \Gamma, P, S)$ jest rekurencyjnie zdefiniowany według następujących reguł:

1. Startujemy z grafu g składającego się z jednego wierzchołka o etykiecie S i numerze porządkowym 1.

2. Jeśli w g istnieje wierzchołek n zaetykietowany przez A oraz produkcja o postaci zredukowanej $A \to a \in P$, to zastępujemy etykietę A przez a.

3. Jeśli w g istnieje wierzchołek n zaetykietowany przez A oraz produkcja o postaci ekspansywnej $A \to a\ x_1 B_1\ x_2 B_2^{e_2} \ldots x_r B_r^{e_r} \in P$, to krok wywodu zgodny z tą produkcją przebiega w następujący sposób:

a) w miejsce wierzchołka n wstawiamy poddrzewo prawej strony produkcji, tzn. wierzchołek zaetykietowany przez A jest zastępowany przez wierzchołek zaetykietowany przez a, który równocześnie dziedziczy po nim numer porządkowy;

b) numer porządkowy l_j każdego wierzchołka n_j etykietowanego przez B_j, $j = 1, \ldots, r$, w generowanym grafie g jest obliczany jako:

$$l_j = l' * p,$$

gdzie l' jest numerem porządkowym wierzchołka n_j w prawej stronie produkcji, zaś p jest numerem porządkowym wierzchołka n w generowanym grafie g;

c) stosujemy transformację osadzenia: załóżmy, że wierzchołek n' w generowanym grafie g:

- jest etykietowany takim samym symbolem nieterminalnym co pewien wierzchołek n_j (etykietowany przez B_j),
- jest oznaczony przez *,
- numer porządkowy wierzchołka n_j jest wielokrotnością numeru porządkowego wierzchołka n',
- nie istnieje krawędź idąca od poprzednika([1]) wierzchołka n' do wierzchołka n_j (patrz przykładowo, rys. 12.3 – wierzchołki $A^*_{[1]}$ i $A^*_{[2]}$).

Wtedy wierzchołek n' usuwamy, a wszystkie krawędzie po nim dziedziczy wierzchołek n_j.

Wprowadźmy pojęcia służące do określenia jednoznacznej reprezentacji scen przez grafy klasy IE [36], która jest podklasą rodziny grafów EDG.

Krawędzie grafów będą używane do definiowania relacji przestrzennych pomiędzy parami obiektów sceny reprezentowanymi przez wierzchołki grafów. Relacje te będą opisywane za pomocą zbioru etykiet krawędziowych Γ, który jest traktowany jako rodzina relacji przeciwsymetrycznych [36]. Dla każdej etykiety $\lambda \in \Gamma$ zdefiniujemy etykietę odwrotną $\lambda^{-1} \in \Gamma$ taką, że dla dowolnych wierzchołków grafu v oraz w reprezentujących obiekty sceny i takich, że $(v, \lambda, w) \in E$, krawędź (w, λ^{-1}, v) opisuje tą samą relację pomiędzy obiektami. Mówimy, że te krawędzie są semantycznie równoważne i zapisujemy

$$(v, \lambda, w) - sem - (w, \lambda^{-1}, v).$$

([1]) Poprzednikiem wierzchołka n' nazywamy wierzchołek n taki, że istnieje krawędź $(n, \beta, n') \in E$.

Na przykład, krawędzie typu: „znajduje się na lewo od" i „znajduje się na prawo od" są semantycznie równoważne. W tak określonym zbiorze Γ wprowadzamy relację liniowego porządku

$$\Gamma = \{\gamma_1, \ldots, \gamma_k \,|\, \gamma_1 \leq \cdots \leq \gamma_k\}.$$

Następnie na zbiór krawędzi Γ nakładamy następujący warunek: dla każdego $v \in V$: jeśli istnieje $(v, \lambda, w) \in E$, to nie istnieje $(v, \gamma, z) \in E$ taka, że $\lambda = \gamma$ i nie istnieje $(z, \beta, v) \in E$, taka, że $\lambda = \beta^{-1}$.

Niech scena P składająca się z n obiektów $K_1, \ldots, K_n$, których położenie jest opisane przez współrzędne kartezjańskie $(x_1, y_1), \ldots, (x_n, y_n)$ będzie reprezentowana przez graf $EDG\, g$. Wierzchołek $v_s \in g$ odpowiadający obiektowi K_s nazywamy *S-wierzchołkiem* jeśli(2)

$$(x_s, y_s) = \min_{y_j}\{\min_{x_i}\{(x_i, y_j)\}\}, \quad i, j = 1, \ldots, n.$$

Graf EDG $H = (V, E, \Sigma, \Gamma, \phi)$ nazywamy *zindeksowanym, krawędziowo-jednoznacznym grafem* (grafem IE – ang. Indexed Edge-unambiguous Graph) [36], jeśli:

1) Γ i E mają własności opisane powyżej;

2) zbiór wierzchołków V jest zindeksowany według następujących reguł:

a) S-wierzchołek – v_0 indeksujemy przez 1,

b) indeksujemy wszystkie wierzchołki przyległe do v_0 zgodnie z relacją $\leq$ w zbiorze etykiet krawędzi łączących v_0 z wierzchołkami przyległymi; indeksacji dokonujemy w porządku rosnącym $i = 2, 3, \ldots, k$ pamiętając, że każdą krawędź wchodzącą do v_0 traktujemy jako krawędź odwrotną wychodzącą z v_0;

c) kolejno wybieramy wierzchołki zindeksowane $i = 2, 3, \ldots, k$ i indeksujemy wierzchołki przyległe do nich, które do tej pory nie zostały zindeksowane;

d) powtarzamy punkt c), aż zindeksujemy wszystkie wierzchołki;

---

(2) Tak zdefiniowany S-wierzchołek odpowiada „obiektowi obrazu, umieszczonemu najniżej z lewej strony". Czasem wygodnie jest przyjąć inną regułę np. „najwyżej umieszczonemu, górnemu obiektowi obrazu". Istotą pojęcia S-wierzchołka jest, aby dało się go w jednoznaczny sposób określić.

3) Zbiór krawędzi E jest przetransformowany tak, że wszystkie krawędzie są skierowane od wierzchołka o niższym indeksie do wierzchołka o wyższym indeksie (gdy zachodzi potrzeba, to „odwracamy" krawędź, tzn. zastępujemy ją przez odwrotną – semantycznie równoważną).

Zdefiniujemy teraz gramatykę grafową *edNLC* [33], dla podklasy której – $ETL(1)$ – został zdefiniowany analizator syntaktyczny [37].

Krawędziowo-etykietowaną, skierowaną, sterowaną etykietą wierzchołka gramatyką grafową (gramatyką $\mathcal{G}_{edNLC}$ – ang. edge-labelled directed Node-Label Controlled graph grammar) nazywamy piątkę

$$\mathcal{G}_{edNLC} = (\Sigma, \Delta, \Gamma, P, Z),$$

gdzie Σ – skończony, niepusty zbiór etykiet wierzchołkowych, Δ (podzbiór Σ) – zbiór terminalnych etykiet wierzchołkowych, Γ – skończony, niepusty zbiór etykiet krawędziowych, P – skończony zbiór produkcji o postaci (l, D, C), gdzie $l \in \Sigma$, D jest grafem EDG, $C : \Gamma \times \{in, out\} \rightarrow 2^{\Sigma \times \Sigma \times \Gamma \times \{in,out\}}$ jest transformacją osadzenia, Z – startowy graf EDG.

Wywód w gramatyce grafowej $\mathcal{G}_{edNLC} = (\Sigma, \Delta, \Gamma, P, Z)$ jest rekurencyjnie zdefiniowany według następujących reguł:

1. Startujemy z grafu $g = Z$.

2. Jeśli w g istnieje wierzchołek nieterminalny n etykietowany przez $l \in \Sigma$ oraz $(l, D, C) \in P$, to zastępujemy wierzchołek n przez graf D.

3. Stosujemy transformację osadzenia C w następujący sposób. Jeśli $(a, c, y, out) \in C(x, in)$, to krawędź etykietowaną przez x $(C(x, \ldots))$, która wchodziła $(C(\ldots, in))$ do wierzchołka n zastępujemy przez krawędź:

a) łączącą wierzchołek grafu D etykietowany przez a $((a, \ldots, \ldots, \ldots))$ z wierzchołkiem rest-grafu([3]) etykietowanym przez $c((\ldots, c, \ldots, \ldots,))$,

b) etykietowaną przez y $((\ldots, \ldots, y, \ldots))$,

c) i wychodzącą z wierzchołka a grafu D $((\ldots, \ldots, \ldots, out))$([4]).

Pojęcia te zostały zdefiniowane w [36].

([3]) Rest-grafem nazywamy wywodzony graf bez wierzchołka n.

([4]) Reguła transformacji osadzenia została podana w intuicyjny sposób. Formalną definicję można znaleźć w [33].

Niech $H = (V, E, \Sigma, \Gamma, \phi)$ będzie grafem IE. Wierzchołek o indeksie 1 nazywamy *wierzchołkiem pierwszego poziomu*.Wierzchołek $v \in V$ nazywamy *wierzchołkiem poziomu* n, jeśli:

1) istnieje krawędź $(w, \lambda, v) \in E$ taka, że w jest poziomu $n-1$,

2) dla każdej krawędzi $(u, \beta, v) \in E$ lub $(v, \alpha, u) \in E$ wierzchołek u jest wierzchołkiem przynajmniej poziomu $n-1$.

Niech piątka $\mathfrak{G}_{edNLC} = (\Sigma, \Delta, \Gamma, P, Z)$ będzie gramatyką grafową klasy $edNLC$. Gramatykę $\mathfrak{G}_{edNLC}$ nazywamy *gramatyką grafową klasy* $L(1)$, jeśli:

1) P jest skończonym zbiorem produkcji o postaci (l, D, C), gdzie

a) $l \in \Sigma$,

b) D jest grafem IE o opisie charakterystycznym:

Y_1	Y_2	...	Y_m, lub	Y_1, gdzie	Y_i jest opisem charakterystycznym
r_1	r_2	...	r_m	0	r_i
E_1	E_2	...	E_m	–	E_i
I_1	I_2	...	I_m	–	I_i

wierzchołka Y_i, $i = 1, \ldots, m$, $Y_1 \in \Delta$, tzn. Y_1 jest wierzchołkiem terminalnym, Y_i jest wierzchołkiem drugiego poziomu,

c) $C : \Gamma \times \{in, out\} \to 2^{\Sigma \times \Sigma \times \Gamma \times \{in,out\}}$ jest transfomacją osadzenia.

2) W zbiorze P nie istnieją dwie produkcje o tej samej lewej stronie i tej samej etykiecie wierzchołka o indeksie 1.

3) Z jest grafem IE o opisie charakterystycznym spełniającym warunek 1b.

Gramatykę grafową $L(1)$ nazywamy *domkniętą gramatyką grafową klasy* $L(1)$, jeśli dla każdego wywodu w tej gramatyce $Z = g_0 \to g_1 \to \ldots \to g_n$, graf g_i, $i = 0, \ldots, n$ jest grafem IE.

Niech $Z = g_0 \to g_1 \to \ldots \to g_n$ będzie wywodem w domkniętej gramatyce grafowej $L(1)$. Wywód nazywamy *regularnym wywodem lewostronnym*, jeśli:

1) dla każdego $i = 0, \ldots, n-1$, w grafie g_i stosujemy produkcję dla wierzchołka mającego najmniejszy indeks,

2) indeksy wierzchołków nie zmieniają się w trakcie analizy.

Niech $\mathfrak{G}_{edNLC} = (\Sigma, \Delta, \Gamma, \mathfrak{P}, Z)$ będzie domkniętą gramatyką grafową $L(1)$. *Potencjalnym kontekstem poprzedzającym* etykiety wierzchołkowej $a \in \Sigma$ nazywamy parę (b, x), $b \in \Delta$, $x \in \Gamma$, jeśli istnieje graf IE $g = (V, E, \Sigma, \Gamma, \phi)$ należący do dowolnego ciągu regularnego wywodu lewostronnego w gramatyce $\mathfrak{G}_{edNLC}$ taki, że

$$(b_k, x, a_l) \in E,$$

gdzie k oraz l są indeksami wierzchołków b_k oraz a_l (odpowiednio).

Domkniętą gramatykę grafową $L(1)$ nazywamy *gramatyką grafową klasy* $ETL(1)$, jeśli dla każdej produkcji

$$\begin{array}{llll} (l)A \rightarrow X_1 & X_2 & \dots & X_m \\ r_1 & r_2 & \dots & r_m \\ E_1 & E_2 & \dots & E_m \\ I_1 & I_2 & \dots & I_m \end{array}$$

zachodzi następujący warunek: jeśli $(b_1, y), \dots, (b_k, y)$ są potencjalnymi kontekstami poprzedzającymi dla A, to istnieje element transformacji osadzenia produkcji (l) o postaci:

$$C(y, in) = \{(X_i, b_1, z_1, in), \dots, (X_i, b_k, z_k, in)\},$$

gdzie $i \in \{1, \dots, m\}$ oraz jeśli $i = 1$, to $z_l = y$, $l \in \{1, \dots, k\}$.

WYKAZ OZNACZEŃ

$\hat{A} : D \to I$ – odwzorowanie opisujące funkcjonowanie algorytmu rozpoznającego

$A : D \to I$ – odwzorowanie przypisujące obiektom indeksy klas

$\mathfrak{A}$ – ogólne oznaczenie automatu

$\alpha \in \mathcal{N}$ – parametr w metodzie αNN

$B : D \to X$ – odwzorowanie przypisujące obiektom wektory cech

β – współczynnik równania regresji

$C : X \to R^L$ – odwzorowanie przypisujące punktom przestrzeni cech wartości funkcji przynależności

D – zbiór rozpoznawanych obiektów

$D^i \subset D$ – podzbiór zbioru D odpowiadający i-tej klasie

$\Delta(x_j)$ – przedział dla cechy x_j w metodzie LI

$\delta_{\mu\nu}$ – funkcja wyboru (delta Kroneckera)

$\varepsilon \in R_+$ – mała dodatnia stała, próg zadziałania

F – funkcja decyzyjna działająca na zbiorze wartości funkcji przynależności $F : R^L \to I \cup \{i^o\}$

F^k – skrótowe oznaczenie numeru klasy, do której zaliczony został k-ty element ciągu uczącego

Φ – zbiór funkcji bazowych $\Phi = \langle \varphi_1, \varphi_2, \ldots \rangle$

$\mathfrak{G}$ – gramatyka formalna $\mathfrak{G} = (\Sigma_N, \Sigma_T, \mathfrak{P}, S)$

$G\left(\frac{x - x^{i,k}}{\eta_k}\right)$ – jądro oszacowania Parzena

H – macierz transformacji liniowej cech o wymiarach $n \times n$

I – zbiór indeksów $I = \{1, 2, \ldots, L\}$

i_o – rozpoznanie neutralne (brak decyzji)

i – numery klas (obrazów) $i = 1, 2, \ldots, L$

j – numer składowej wektora x ; $j = 1, 2, \ldots, n$

$K \subset D \times D$ – relacja klasyfikacji wyodrębniająca w zbiorze D obrazy D^i

$K(\underline{x}, \underline{x}^{i,k})$ – funkcja potencjalna

$\mathfrak{L}$ – język opisany gramatyką $\mathfrak{G}$

$L \in \Xi$ – liczba klas

$\lambda_\nu \in R$ – mnożnik normalizujący, związany za składową x_ν wektora x

λ – pusty napis (w gramatykach formalnych)

M^i – obiekt modalny i-tej klasy $\underline{M}^i = \langle m_1^i, m_2^i, \ldots, m_n^i \rangle$

N – zbiór liczb naturalnych

n – liczba cech (wymiar przestrzeni X)

$N = \#U$ – liczebność ciągu uczącego

$N^i = \#U^i$ – liczba obiektów ciągu uczącego, należących do i-tej klasy

$\mathfrak{P}$ – zbiór produkcji w gramatyce formalnej $\mathfrak{G}$

$Prawd(\omega)$ – prawdopodobieństwo określonego zdarzenia ω

p^i – aprioryczne prawdopodobieństwo pojawienia się obiektu należącego do klasy $i\, p^i = Prawd(d \in D^i)$

$p(\underline{i}/z)$ – prawdopodobieństwo, że obiekt opisany wektorem $\underline{x}$ należy do klasy i

$p(\underline{x}/i)$ – prawdopodobieństwo warunkowe wystąpienia wektora x przy założeniu, że obiekt należy do klasy i

$q_{\mu\eta}$ – funkcja strat związanych z przyjęciem decyzji η w sytuacji, gdy rzeczywista przynależność obiektu jest μ

$Q^i(\underline{x})$ – oczekiwana wartość straty ponoszonej przy wskazaniu klasy i dla obiektu x

$Q(\underline{x})$ – średnia oczekiwana wartość straty ponoszonej przy klasyfikacji obiektu x

$Q(A, \hat{A})$ – ocena jakości algorytmu rozpoznającego $\hat{A}$ w zadaniu rozpoznawania określonym przez A

R – zbiór liczb rzeczywistych

R_* – zbiór liczb nieujemnych

R_+ – zbiór liczb dodatnich

ρ – miara odległości w przestrzeni X ($\rho : X \times X \to R_+$)

S – symbol startowy w gramatyce $\mathfrak{G}$ ($S \in \Sigma_N$)

Σ – alfabet (w gramatykach formalnych)

Σ^+ – zbiór niepustych ciągów elementów alfabetu Σ

Σ^* – zbiór ciągów elementów alfabetu ($\Sigma^* = \Sigma^+ \cup \{\lambda\}$)

Σ_N – zbiór symboli nieterminalnych,

Σ_T – zbiór symboli terminalnych ($\Sigma = \Sigma_N \cup \Sigma_T$ oraz $\Sigma_N \cap \Sigma_T = \emptyset$)

$t \in R$ – wykładnik w metryce Minkowskiego

$\mathbf{T}$ – macierz kowariancji cech o wymiarach $[n \times n]$

$\mathbf{T}^i$ – macierz kowariancji dla i-tej klasy

U – ciąg uczący $U = \{\langle \underline{x}^k, i^k \rangle\}$

$U^i = \{\underline{x}^{i,k}\}$ – podzbiór ciągu uczącego U dla i-tej klasy; zbiór takich $\underline{x}^k$, dla których $i^k = i$

U^α – podzbiór najbliższych obiektów ciągu uczącego wykorzystywany w metodzie αNN

$U^{\alpha,i}$ – podzbiory zbioru U^α

V_ν^i – współczynnik wagowy

W^i – wzorzec i-tej klasy

X – przestrzeń cech

$\underline{x}$ – punkt w przestrzeni X odpowiadający obiektowi $d \in D$ $\underline{x} = \langle x_1, x_2, \ldots, x_n \rangle$ ($\underline{x} = B(d)$)

x_j^k – j-ta składowa wektora $\underline{x}^k$, gdzie k jest numerem obiektu według pewnej ustalonej numeracji

\$ – symbol końca ciągu

Oznaczenia pomocnicze (operacje, operatory, symbole działań)

μ, ν, η – oznaczenia posiłkowe (pomocnicze) definiowane doraźnie i funkcjonujące wyłącznie w ramach jednego wzoru

$|\mu|$ – wartość bezwzględna liczby μ

$|\underline{\mu}|$ – długość wektora μ obliczana jako

$$|\underline{\mu}| = \sqrt{\sum_{nu=1}^{n} (\mu_\nu)^2}$$

$ent(\mu)$ – część całkowita liczby rzeczywistej μ

$E\{\zeta\}$ – wartość oczekiwana (średnia) elementów zbioru Ξ ($\zeta \in \Xi$)

Ξ – ogólny symbol pewnego zbioru

ζ – konkretny element pewnego zbioru (służy do objaśniania notacji; w zastosowaniach zawsze jest zastępowany konkretnym oznaczeniem łacińskim, określającym ustalony zbiór)

$\{\zeta, \ldots\}$ – nawiasy obejmujące elementy zbioru

$\#\Xi$ – moc (liczba elementów) zbioru Ξ

$\emptyset$ – symbol zbioru pustego

$\langle \zeta^1, \zeta^2, \ldots \rangle$ – nawiasy wyznaczające zbiór uporządkowanych elementów

ψ – ogólny symbol kwantyfikatora lub zespołu kwantyfikatorów

$\forall$ – kwantyfikator ogólny („dla każdego ...")

$\exists$ – kwantyfikator szczegółowy („istnieje ...")

ω – ogólny symbol wyrażenia logicznego

$\wedge$ – operator logiczny – koniunkcja

$\vee$ – operator logiczny – alternatywa

$\neg$ – operator logiczny – negacja

$\Rightarrow$ – operator logiczny – implikacja

$\equiv$ – operator logiczny – identyczność

$\psi[\omega]$ – stwierdzenie, że prawidłowość określona wyrażeniem logicznym ω zachodzi dla warunków określonych kwantyfikatorem ψ

θ – ogólny symbol operatora

$\zeta^{\theta} \in \Xi$ – symbol operacji θ wykonywanej na wszystkich elementach ζ zbioru Ξ

$\times$ – operator iloczynu kartezjańskiego

$\sum$ – operator sumowania algebraicznego

$\prod$ – operator iloczynu algebraicznego

$\cup$ – operator sumy zbiorów

$\cap$ – operator iloczynu zbiorów

BIBLIOGRAFIA

1. Turbovicz I.T. *Ob optimalnom metode opoznavanija obrazov pri vzaimokorelirovanych priznakach. Opoznavanie obrazov – teorija peredaczi informacii.* Moskva: 1965.
2. Turbovicz I.T. *Opoznavanije obrazov.* Moskva: Nauka 1965.
3. Sobczak W., Malina W. *Metody selekcji informacji.* Warszawa: WNT 1978.
4. Tadeusiewicz R.: *Przykłady wykorzystania metod cybernerycznych. Metody cyfrowej analizy sygnałów wibroakustycznych.* Wrocław: Ossolineum 1979, s. 225-269.
5. Ajzerman M.A., Brawerman E.M., Rozonoer L.I. *Rozpoznawanie obrazów – metoda funkcji potencjalnych.* Warszawa: WNT 1976.
6. Tadeusiewicz R., Kot. L., Mikrut Z. *Biocybernetyka.* Skrypt AGH, Kraków 1982.
7. Nilsson N. *Maszyny uczące się.* Warszawa: PWN 1968.
8. Robbins H., Monro S. *A stochastic appriximation method.* Ann. Math. Statistics 1951, vol. 22.
9. Greblicki W. *Asymptotycznie optymalne algorytmy rozpoznawania i identyfikacji w warunkach probabilistycznych.* Prace ICT Pol.Wrocł., nr 18, seria: Monografie, nr 3, Wrocław 1974.
10. Rozin B.B. *Teoria rozpoznawania obrazów w badaniach ekonomicznych.* Warszawa: PWN 1979.
11. Duda O., Hart P. *Pattern classification and scene analysis.* New York: J.Wiley 1973.
12. Vasilev V.I. *Raspoznajuszczije sistemy.* Kiev: Naukova Dumka 1983.
13. Tadeusiewicz R. *Rozpoznawanie obrazów – zarys teorii.* Skrypt uczelniany UJ, Kraków 1985.
14. Freeman H. *On the encoding of arbitrary geometric configurations,* IEEE Trans. Electron. Comput. EC-10, 1961.

15. Freeman H. *On the digital-computer classification of geometric line patterns*, Proc. Natl. Electron. Conf. vol. 18, 1962.
16. Shaw A.C. *The formal description and parsing of pictures*, SLAC Rep. nr 84, Stanford Linear Accelerator Center, Stanford, Calif. 1968.
17. Jakubowski R. *Extraction of shape features for syntactic recognition of mechanical parts*, IEEE Trans. Syst., Man Cybern. 1985, SMC-15, vol. 5.
18. Jakubowski R. *A structural representation of shape and its features.* Inf. Sci., 1986, vol. 39, nr 2.
19. Fu K.S. *Syntactic Pattern Recognition and Applications.* New Jersey: Prentice–Hall, 1982.
20. Kulikowski J. L. *Zarys teorii grafów.* Warszawa: PWN 1986.
21. Chomsky N. *Syntactic Structures.* Hague: Mouton 1957.
22. Lewis P.M., Stearns R.E. *Syntax-directed transductions*, Journ. ACM 1968, nr 15·
23. Rosenkrantz D.S., Stearns R.E. *Properties of deterministic top down grammars*, ACM Symposium on Theory of Computing 1969.
24. Harrison M.A. *Introduction to Formal Language Theory.* Reading, Massachusetts: Adison–Wesley 1978.
25. Aho A.V., Ullman J.D. *The Theory of Parsing, Translation, and Compiling. Vol. 1, Parsing.* Englewood Cliffs, N.J.: Prentice–Hall 1972.
26. Jakubowski R. *A self-learning system for structural recognition of complex shapes*, Podstawy Sterowania 1987, vol. 17, nr 1–2.
27. Knuth D.E. *On the translation of languages from left to right*, Information Control 1965, nr 8.
28. Floyd R.W. *Syntactic analysis and operator precedence*, Journ. ACM 1963, nr 10.
29. Brainerd W.S. *Tree generating regular systems*, Inf. Control 1969, nr 14.
30. Doner J.E. *Tree acceptors and some of their applications*, Journ. Comp. System Sci. 1970, nr 4.
31. Fu K.S., Kunii T.L. *Picture Engineering.* New York: Springer-Verlag 1982.
32. Lu S.Y., Fu K.S. *A syntactic approach to texture analysis*, Comp. Graph. & Image Processing 1978, nr 7.

33. Janssens D., Rozenberg G., Verraedt R. *On sequential and parallel node-rewriting graph grammars*, Comp. Graph. Image Processing 1983, nr 18.

34. Brandenburg F.J. *On the complexity of the membership problem of graph grammars*, Proc. of the WG'83, International Workshop on Graph-theoretic Concepts in Computer Science, June 16-18 1983, Osnabrück, FRG, Trauner Verlag 1983.

35. Shi Q.Y., Fu K.S. *Parsing and translation of (attributed) expansive graph languages for scene analysis*, IEEE Trans. Patt. Anal. Mach. Intell. 1983, vol. PAMI-5, nr 5.

36. Flasiński M. *Characteristics of edNLC – graph grammar for syntactic pattern recognition*, Comp. Vision, Graph. Image Processing 1989, nr 42.

37. Flasiński M. *Parsing of edNLC – graph grammar for scene analysis*, Patt. Recognition 1988, vol. 21, nr 6.

38. James M. *Classification Algorithms*. New York: John Wiley 1985.

39. Fukunaga K. *Introduction to Statistical Pattern Recognition*. New York: Academic Press 1972.

40. Marek T., Noworol Cz. *Analiza sekwencyjna w badaniach empirycznych*. Warszawa: PWN 1987.

41. Fu K.S. *Sequential Methods in Pattern Recognition and Machine Learning*. New York: Academic Press 1968.

42. Fu K.S. *Digital Pattern Recognition*. Berlin: Springer Verlag 1976.

43. Sebestyen G.S. *Decision in Pattern Recognition*. New York: Macmillan 1962.

44. Nilsson M.J. *Learning Machines – Foundations of Trainable Pattern Classifying Systems*. New York: McGraw-Hill 1965.

45. Mendel J.M., Fu K.S. *Adaptive, Learning and Pattern Recognition Systems – Theory and Applications*. New York: Academic Press 1970.

46. Meisel W. *Computer Oriented Approaches to Pattern Recognition*. New York: Academic Press 1972.

47. Patrick E.A. *Fundamentals of Pattern Recognition*. Princeton: Prentice-Hall 1972.

48. Andrews H.C. *Introduction to Mathematical Techniques in Pattern Recognition*. New York: Willey 1972.

49. Chen C.H. *Statistical Pattern Recognition*. Washington, D.C.: Hayden Book Co. 1973.
50. Tadeusiewicz R. *Biocybernetyka*. Wrocław: Ossolineum 1988.
51. Tadeusiewicz R. *W stronę uśmiechniętych maszyn*. Warszawa: Alfa 1989.
52. Tadeusiewicz R. *Podstawy biocybernetyki*. Warszawa: PWN 1991.
53. Tadeusiewicz R. *Hierarchiczny system oprogramowania systemu analizy i rozpoznawania obrazów*, Przetwarzanie sygnałów w telekomunikacji, sterowaniu i kontroli, Bydgoszcz 1984.
54. Tadeusiewicz R. *Komputerowa analiza obrazów i jej zastosowania*, Elektrotechnika 1982, tom 1, zeszyt 2.
55. Tadusiewicz R. *Systemy wizyjne dla robotów przemysłowych: rola, budowa zastosowanie*, Zeszyty Naukowe AGH, Automatyka 1989, nr 47.
56. James M. *Pattern Recognition*. New York: John Wiley 1988.
57. Tadeusiewicz R. *Wybrane zagadnienia rozpoznawania obrazów dźwiękowych*. Rozprawa doktorska, AGH, Kraków 1975.
58. Tadeusiewicz R. *Ocena przydatności wybranych metryk w minimalnoodległościowych metodach rozpoznawania mowy*, Archiwum Akustyki 1983, t. 18, nr 3, s.275–284.
59. Kot. L. *Ocena przydatności analizy pasmowej do rozpoznawania prostych elementów mowy polskiej przez maszynę cyfrową*. Rozprawa doktorska, AGH, Kraków 1980.
60. Tadeusiewicz R. *Rozpoznawanie obrazów w zastosowaniach ekonomicznych*. Kraków: Akademia Ekonomiczna 1985.
61. Izworski A. *Globalna metoda segmentacji zredukowanego widma sygnału mowy*. Rozprawa doktorska, AGH, Kraków 1986.
62. Borysiewicz J., Tadeusiewicz R. *Computer evaluation of antiviral activities of some thiosemicarbozones in experiments in vivo*, Acta Virologica 1976, nr 20, s. 402–410.
63. Rumelhart D. E., McClelland J. L. (eds) *Parallel Distributed Processing: Explorations in Microstructures od Cognition*, vol. 1, Cambridge: MIT Press 1986.
64. Międzobrodzki J., Tadeusiewicz R., Heczko P. B. *Virulence of Coagulasonegative Staphylococci for Chick Embryos*, The Staphylococci, Zbl. Bakt. Suppl. 14, Stuttgart--New York: Gustaw Fisher Verlag 1985, s.447–481.

W serii Współczesna Nauka i Technika ukazały się książki

Nowe Materiały i Technologie

- **Z. Bojarski, H. Morawiec**
  Metale z pamięcią kształtu
- **R. Pampuch, S. Błażewicz, J. Chłopek, A. Górecki, W. Kuś**
  Nowe materiały węglowe w technice i medycynie
- **W. Włosiński**
  Spajanie metali z niemetalami

Informatyka

- **L. Bolc, J. Cytowski**
  Metody przeszukiwania heurystycznego t. 1, t. 2
- **Z. Bubnicki**
  Wstęp do systemów ekspertowych
- **Z. Jurkiewicz, M. Lao**
  Język programowania LISP